НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Автоматики та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
"04" грудня 2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації)_____121 Інженерія програмного забезпечення_____
(код і назва спеціальності)

на тему: Платформа для розроблення програмного забезпечення позаштатними працівниками

Виконав (-ла): студент (-ка) 6 курсу, групи ІТ-73мп
(шифр групи)

Тимошенко Олександр Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент, Катін П.Ю _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

АНОТАЦІЯ

Магістерська дисертація містить 97 сторінки, 47 рисунків, 19 таблиць, список літератури з 29 найменувань та 10 додатків.

Ключові слова: програмне забезпечення, платформа для розроблення програмного забезпечення, веб-додаток, база даних, програмний сервіс.

Метою магістерської дисертації є проектування та розробка платформи для розроблення програмного забезпечення, що реалізує концепцію програмного сервісу з урахуванням позаштатності працівників, по якому було опубліковано статтю з назвою «Концепція сервісу програмного забезпечення», на IV Міжнародній науково-практичній конференції Summer InfoCom Advanced Solutions 2017.

В результаті дослідження було створено платформу що об’єднує всі етапи життєвого циклу програмного забезпечення, містить функціонал пошуку працівників, керування процесом розроблення, оплати послуг.

ANNOTATION

The master's dissertation contains 97 pages, 47 figures, 19 tables, a list of literature of 29 titles and 10 applications.

Keywords: software, software development platform, web application, database, software service.

The purpose of the master's thesis is to design and develop a platform for the development of software that implements the concept of a software service, taking into account the freelance workforce, on which the article entitled "The Concept of Software Services" was published at the IV International Scientific and Practical Conference Summer InfoCom Advanced Solutions 2017.

As a result of the study, a platform was created that unites all stages of the software lifecycle, contains a functional search for employees, development process management, payment services.

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) Інформатики та обчислювальної техніки _____
(повна назва)

Кафедра Автоматики та управління в технічних системах _____
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 121 Інженерія програмного забезпечення _____
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Тимошенку Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Платформа для розроблення програмного забезпечення позаштатними працівниками _____
_____
_____
науковий керівник дисертації Катін Павло Юрійович_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «___»_____ 2018 р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження процес розроблення програмного забезпечення позаштатними працівниками _____
_____

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) платформа для розроблення програмного забезпечення позаштатними працівниками , концепція сервісу програмного забезпечення _____
_____

5. Перелік завдань, які потрібно розробити аналіз існуючих платформ для розроблення програмного забезпечення позаштатними працівниками, характеристика платформи для розроблення програмного забезпечення позаштатними працівниками, проектування платформи для розроблення програмного забезпечення позаштатними працівниками, стартап проект

6. Орієнтовний перелік ілюстративного (графічного) матеріалу концептуальна модель бази даних, діаграма бази даних, діаграма прецедентів, діаграма послідовності, структурна схема, діаграма компонентів, діаграма класів, діаграма розгортання

7. Орієнтовний перелік публікацій концепція сервісу програмного забезпечення

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 29 жовтня

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Огляд і аналіз існуючих рішень	29.10.18 – 07.11.18	
2	Вибір технологій	07.11.18 – 09.11.18	
3	Проектування архітектури системи	09.11.18 – 12.11.18	
4	Проектування бази даних	12.11.18 – 14.11.18	
5	Узгодження спроектованої системи	14.11.18 – 15.11.18	
6	Розроблення серверної частини	15.11.18 – 20.11.18	
7	Розроблення клієнтської частини	20.11.18 – 24.11.18	
8	Тестування системи	24.11.18 – 29.11.18	
9	Оформлення пояснювальної записки	29.11.18 – 04.12.18	
10	Подача дисертації на перевірку	04.12.18	

Студент _____ Тимошенко О.О.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Катін П.Ю.

* Консультантом не може бути зазначено наукового керівника

(підпис) (ініціали, прізвище)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HTTP – Hyper Text Transfer Protocol

IoC – Inversion of Control

ORM – Object-relational mapping

REST – Representational State Transfer

RPC – Remote procedure call

SaaS – Software as a service

TCP – Transmission Control Protocol

АРМ – автоматизоване робоче місце

ПЗ – програмне забезпечення

СКБД – система керування базами даних

СПЗ – сервіс програмного забезпечення

ВСТУП

В сучасних умовах досить значна кількість галузей потребують втручання з боку сфери інформаційних технологій для зіставлення рівня конкурентоспроможності на ринку[1-6]. Також більшість стартап-проектів не обходяться без участі сфери інформаційних технологій, а на сьогоднішній день розроблення програмного забезпечення є складним процесом, що може включає в себе багато інших процесів таких як:

- формування вимог;
- проектування;
- реалізація;
- тестування;
- впровадження;
- експлуатація та супровід.

Також багато інших в залежності від предметної області і щоб отримати якісне програмне забезпечення, не витративши при цьому надвелику кількість коштів та часу, потрібно добре розбиратися в процесах розроблення програмного забезпечення, що зазвичай є проблемою для більшості споживачів послуг розроблення програмного забезпечення, тому для вирішення даної проблеми є два варіанти:

1) віддати свій проект в компанію що займається розробленням програмного забезпечення на замовлення;

2) найняти потрібних людей і керувати розробленням проекту самостійно.

Перший варіант задовольнить тих, в кого є значний бюджет. Другий варіант потребує менших витрат коштів, але більших витрат часу, що в результаті може призвести до більших витрат ніж перший варіант через недосвідченість в галузі розроблення програмного забезпечення, а то й взагалі згубити проект через брак часу. Але й перший варіант має недоліки, навіть володіючи значним бюджетом, на

виході можна отримати зовсім інше програмне забезпечення ніж бажане.

Тому ринок потребує нових рішень в галузі створення програмного забезпечення. Одне з них, а саме «Програмний сервіс», було запропоноване як, «Сервіс безперервної розробки ПЗ (що має замінити собою весь життєвий цикл ПЗ: планування, розробку, тестування, модифікацію, супровід, оплату, тощо)» [1-19]. З його допомогою клієнт, що не розуміється на процесах розроблення програмного забезпечення, зможе керувати процесом розроблення, використовуючи вже існуючі перевірені робочі процеси розроблення ПЗ, або використовуючи ресурси знайдених через програмний сервіс людей, навіть без необхідного досвіду, отримавши при цьому потрібний йому програмний продукт.

Але «Сервіс безперервної розробки ПЗ» є лише концепцією, яку буде реалізовано в даній роботі, як платформу для розроблення програмного забезпечення. А для того щоб звузити і так величезну кількість процесів що стосуються розроблення ПЗ у великих компаніях, «Сервіс безперервної розробки ПЗ» буде реалізовуватися для позаштатних працівників, для яких можна створити робочий процес розроблення ПЗ, результатом використання якого, замовник отримає якісне ПЗ незважаючи на позаштатність працівників, за умови дотримання робочого процесу платформи.

Актуальність даної дослідної роботи полягає у необхідності вирішення проблеми щодо об’єднання всіх етапів розроблення ПЗ в єдиний сервіс для надання послуг по розробленню ПЗ, в контексті позаштатних працівників

Мета дослідження полягає у підвищенні якості та ефективності процесу надання програмних послуг, що безпосередньо охоплює стандартний життєвий цикл програмного забезпечення (планування, розробку, тестування, модифікацію, супровід) та певні підтримуючі процеси (оплату, інтеграцію, оренду, взаємодію клієнта та провайдера), виявлення переваг та недоліків, у порівнянні із аналогами, обґрунтування доцільності й ефективності застосування платформи для розроблення програмного забезпечення позаштатними працівниками.

Об'єктом дослідження є процес розроблення програмного забезпечення позаштатними працівниками.

Предметом дослідження є платформа для розроблення програмного забезпечення позаштатними працівниками та сервіс програмного забезпечення.

1 АНАЛІЗ ІСНУЮЧИХ ПЛАТФОРМ ДЛЯ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Як таких «Платформ для розроблення програмного забезпечення» на даний момент немає, але розробляєму платформу можна порівняти з сучасними системами управління проектами, такими як JIRA[1], певні спільні характеристики має навіть веб-сервіс для спільної розробки програмного забезпечення GitHub[2], оскільки поєднує в собі систему контролю версій Git та власну реалізацію системи управління проектами, останнім часом на ринку з’являються новітні системи управління проектами такі як Comindware[3], що мають так звану автоматизацію більшості процесів розроблення програмного забезпечення. Але найблищим по схожості до платформи для розроблення програмного забезпечення є системи що надають послуги позаштатних працівників, такі як upwork.

1.1 Система відстежування помилок JIRA

JIRA – це система відстежування помилок, але зазвичай її використовують як систему для керування проектами[1]. Система була запущена в 2002. Розробник компанія Atlassian надає безкоштовну ліцензію для проектів з відкритим вихідним кодом та для навчальних проектів, Проект був реалізований на технології Java, відповідно для використання потрібна система із встановленою віртуальною машиною.

Це одна з найдавніших та найпопулярніших систем для відстежування помилок, що надає їй багато переваг, отриманих за період використання користувачами. Має сумісність з різними системами версійного контролю, що дозволяє переглядати інформацію про всі дії з програмним кодом, що проведені над задачею і відслідковувати зміни ієрархії задач.,

Однією з ключових можливостей JIRA є можливість змінювання робочого

процесу (Custom Workflow), стандартний робочий процес можна подивитись на рисунку 1.1.

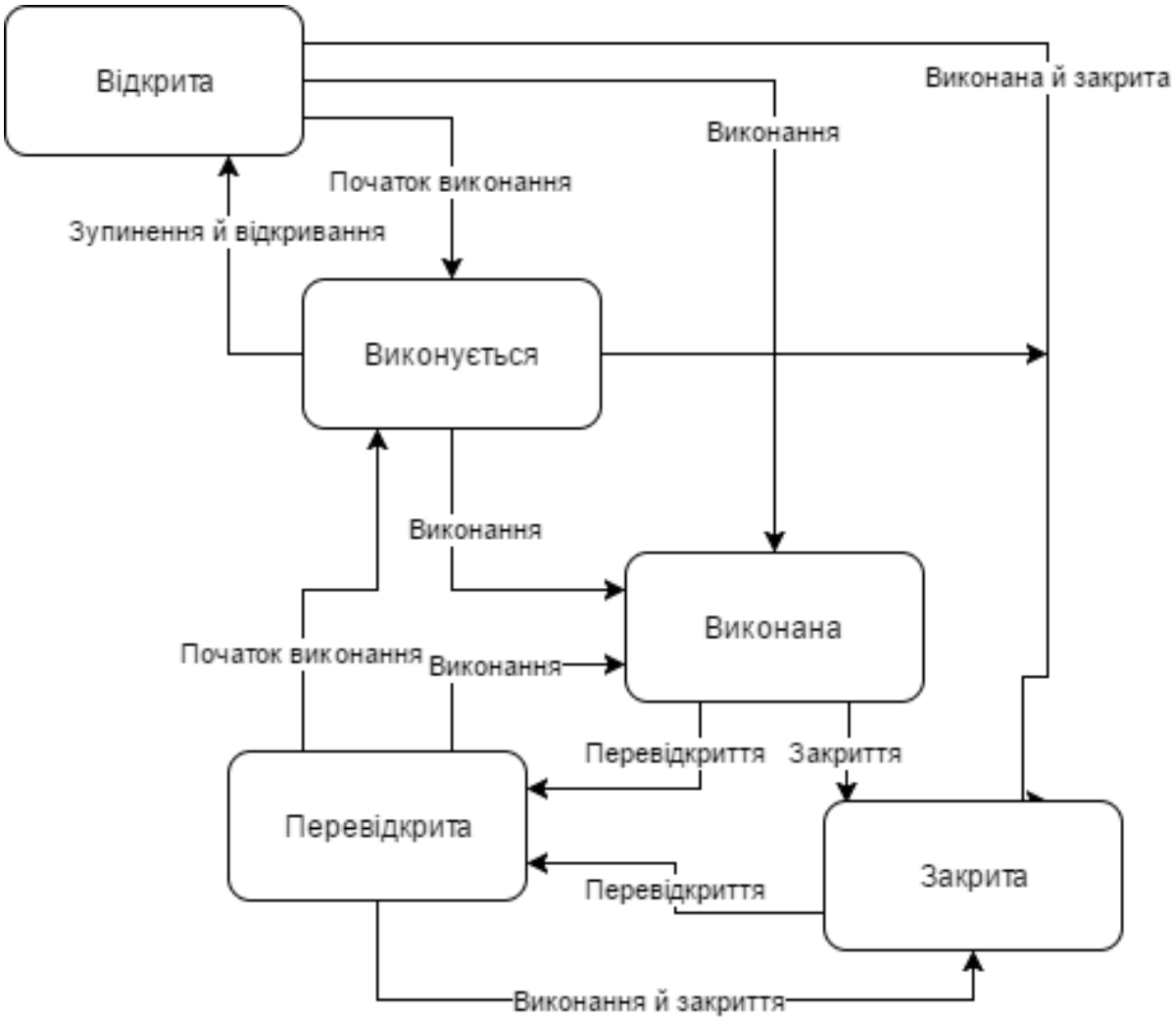


Рисунок 1.1 – Робочий процес JIRA

Також JIRA має можливість змінювати робочий процес відповідно до потреб користувача, так наприклад можна зробити обов’язковим процес огляду коду, або взагалі прибрати його з робочого процесу розроблення ПЗ. Але є й недоліком цієї можливості є досить складний процес зміни робочого процесу, для внесення змін потрібно володіти навичками розробника ПЗ, або розумітися на скриптових мовах програмування, оскільки для внесення змін потрібно знати, як працювати з специфічною для JIRA скриптовою мовою програмування.

JIRA має веб-додаток, який зображено на рисунку 1.2, де можна переглянути

інформацію про задачу, додати коментарі, змінити статус задачі, змінити пріоритет, прикріпити файли, подивитися історію змін, будь-яку активність над задачею, змінити виконавця.

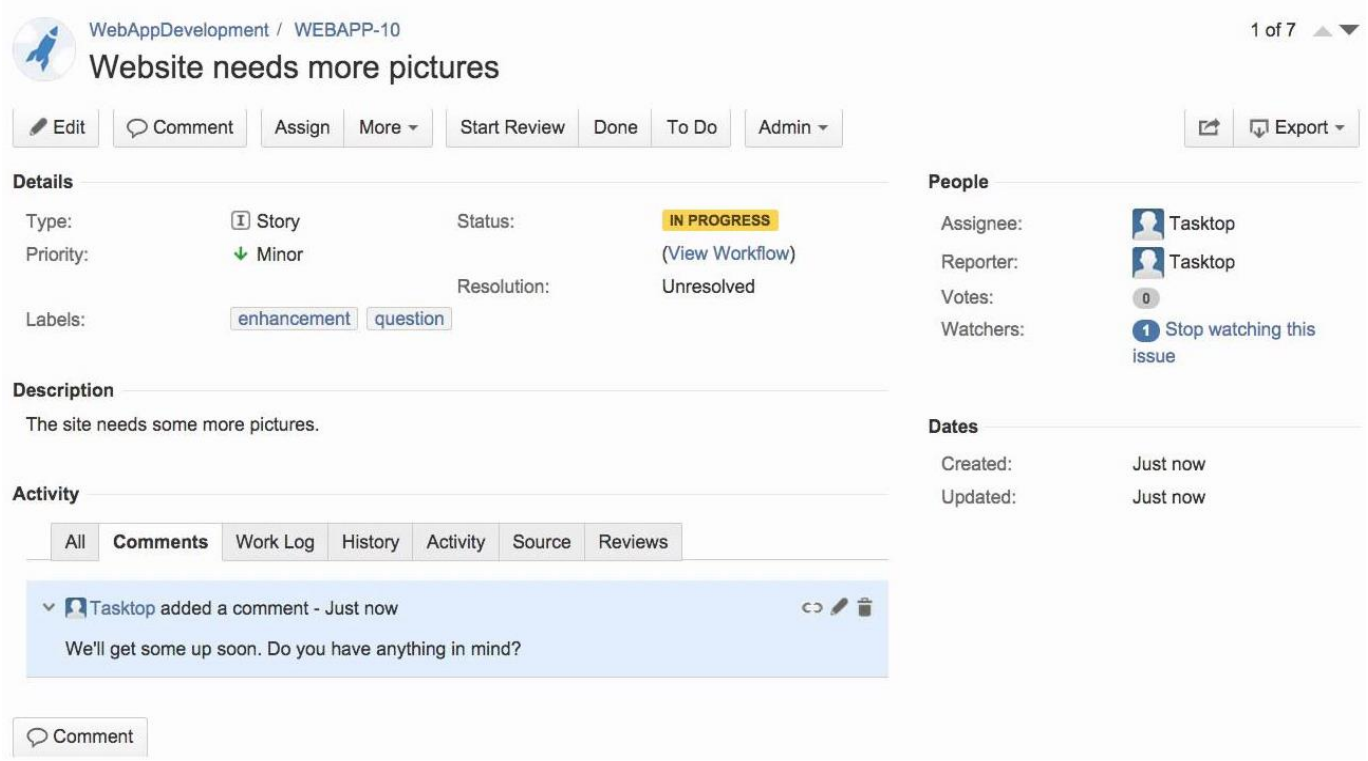


Рисунок 1.2 – Веб-додаток JIRA[1]

1.2 Система управління проектами Comindware

Організація процесу розробки програмного забезпечення – складний процес, для управління яким не завжди підходять системи відстеження запитів або управління проектами. Як правило, існує значний обсяг робіт, що вимагає обробки за допомогою гнучких у конфігурації бізнес-процесів, що дозволяють налаштовувати переходи між завданнями, затверджувати і контролювати їх, взаємодіяти в єдиній системі. Рішення Comindware для управління розробкою ПЗ дозволяє налаштовувати і автоматизувати бізнес-процеси розробки програмного забезпечення, такі як відстеження запитів і помилок, управління змінами і узгодженням. Рішення також забезпечує управління процесами відстеження показників ефективності (KPI) і відповідності умов угод (SLA)[3].

Система управління розробкою ПЗ Comindware об'єднує всі бізнес-процеси, а також пов'язані з ними обговорення, дані і документи в єдиному інтуїтивно зрозумілому інтерфейсі представленому на риснку 1.3, що значно полегшує командну взаємодію.

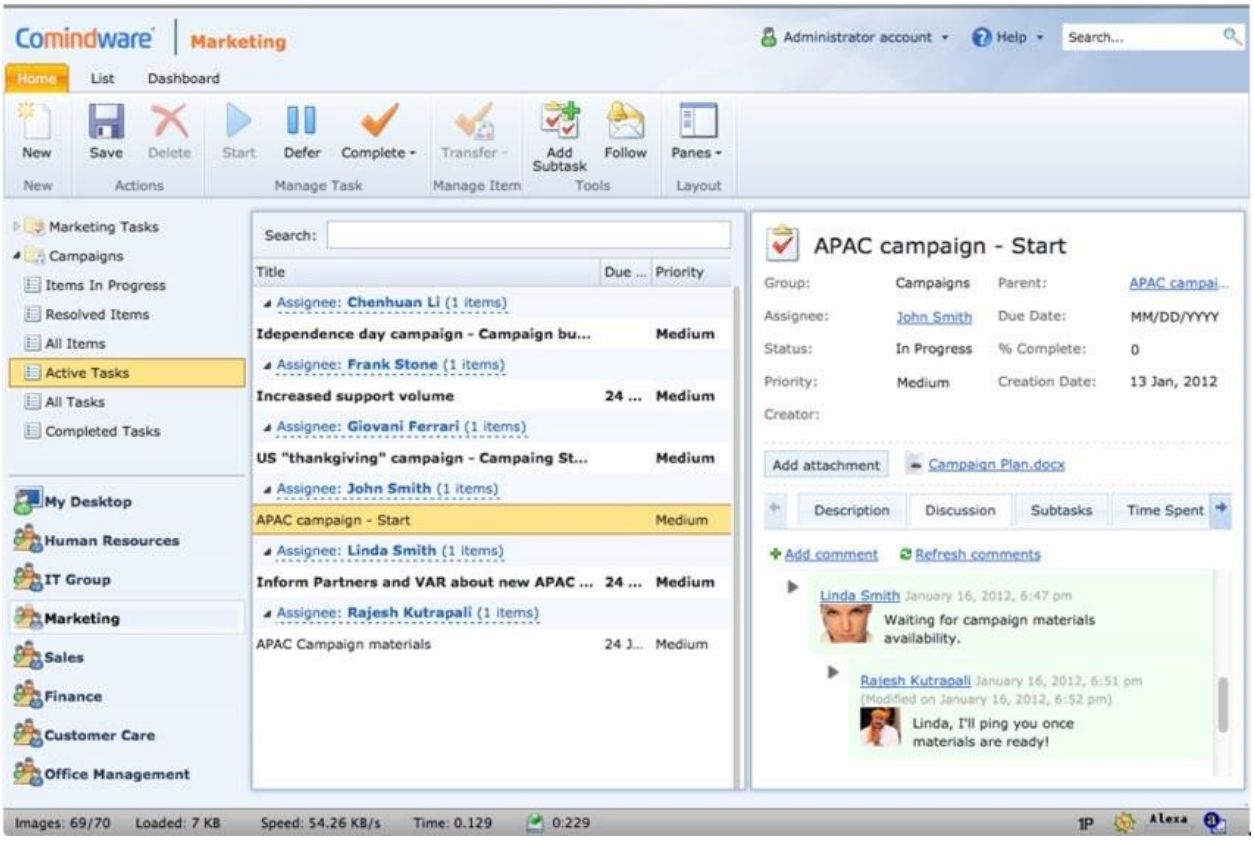


Рисунок 1.3 – Графічний інтерфейс Comindware

Серед них задач, вирішуваних системою управління розробкою ПЗ Comindware, є такі процеси, як:

- відстеження помилок;
- управління запитами на зміну умов;
- управління інцидентами і проблемами;
- відстеження виконання умов контрактів;

Бізнес-проблеми, виявлені в процесі розробки:

- непрозорість повсякденних бізнес-процесів;
- нечіткий розподіл завдань;

– неструктуровані дані і документи.

Переваги використання Comindware для управління розробкою ПЗ:

– централізоване управління ресурсами, даними і завданнями;

– контроль статусів завдань в реальному часі;

– оптимізація процесів узгодження;

– підвищення продуктивності при роботі в команді.

В процесі розробки відповідне програмне забезпечення передбачає організацію процесу через наступні етапи:

1) планування: поділ проекту на етапи і формування календарного плану. Узгодження рівня документування і тестування;

2) призначення: гарантування терміну виконання, погодження з замовником фахівців. Призначені програмісти надійно закріплюються на проекті;

3) щоб уникнути помилок і їх виправлення – постійний контроль за ходом проекту;

4) оцінка наслідків та ефективності рішень. Проведення експериментів перед прийняттям технології на озброєння;

5) удосконалення: аналіз виконаних операцій, який переслідує мету вдосконалення;

6) перевірка якості.

Особлива увага приділяється тестуванню. Причому, виконавши внутрішнє тестування, надається замовнику можливість провести власну перевірку до прийняття. Факторами створення якісного продукту є:

– кваліфікована команда;

– досвідчені менеджери проектів;

– безперервне навчання співробітників;

– спланований і контрольований процес розроблення ПЗ;

– документування виробництва;

– обов'язкове і професійне тестування;

– тестування за допомогою авто-тестів;

– прагнення підвищити ефективність системи.

Система досить інноваційна в своєму роді, як для системи керування проектами, оскільки має багато визначених процесів розроблення ПЗ, якими дозволяє користуватися.

1.3 Система пошуку позаштатних працівників UpWork

Стосовно залучення до процесу розробки позаштатних працівників найбільш схожими існуючими рішеннями є ресурси для пошуку позаштатних працівників є такі як “ UpWork” [20], але ці ресурси не є специфічними для розроблення програмного забезпечення, там розміщуються завдання і виконавці завдань, а робочий процес виконання завдань розробляється замовником та виконавцем під час виконання замовлення. Для кожного завдання по розробленню програмного забезпечення розробляється новий робочий процес, який зазвичай не є ефективним через постійне витрачання часу на створення нового робочого процесу, причому зазвичай робочий процес створюється недосвідченими працівниками в результаті чого програмний продукт може бути недостатньо якісним, оскільки зазвичай таке розроблення програмного забезпечення оминає процеси тестування, формування документації, огляду коду іншими працівниками для покращення його якостей. UpWork є найбільш схожим на платформу розроблення програмного забезпечення рішенням оскільки тут є і управління проектами, і пошук працівників, і оплата послуг, але відповідно управління проектами відбувається через листування замовника і виконавця, і немає єдиного визначеного робочого процесу, якість отримуємих послуг на низькому рівні оскільки багато етапів розроблення програмного забезпечення оминається, замовних хоче зекономити, а виконавець зробити побільше і пошвидше. Система представляє веб-додаток інтерфейс якого представлено на рисунку 1.4. Інтерфейс додатку досить сучасний і використовує адаптивний інтерфейс, який змінюється в залежності від розмірів екрану, тому це дозволяє системі працювати одночасно добре на мобільних пристроях та на персональних комп’ютерах, але відповідно супроводжувати клієнтську частину коду

в такому випадку стає складніше, ніж розроблення паралельних версій для комп'ютерів та для мобільних пристроїв. Але сам по собі інтерфейс дуже дружелюбний до користувачів, як на мобільних пристроях так і на персональних компютерах.

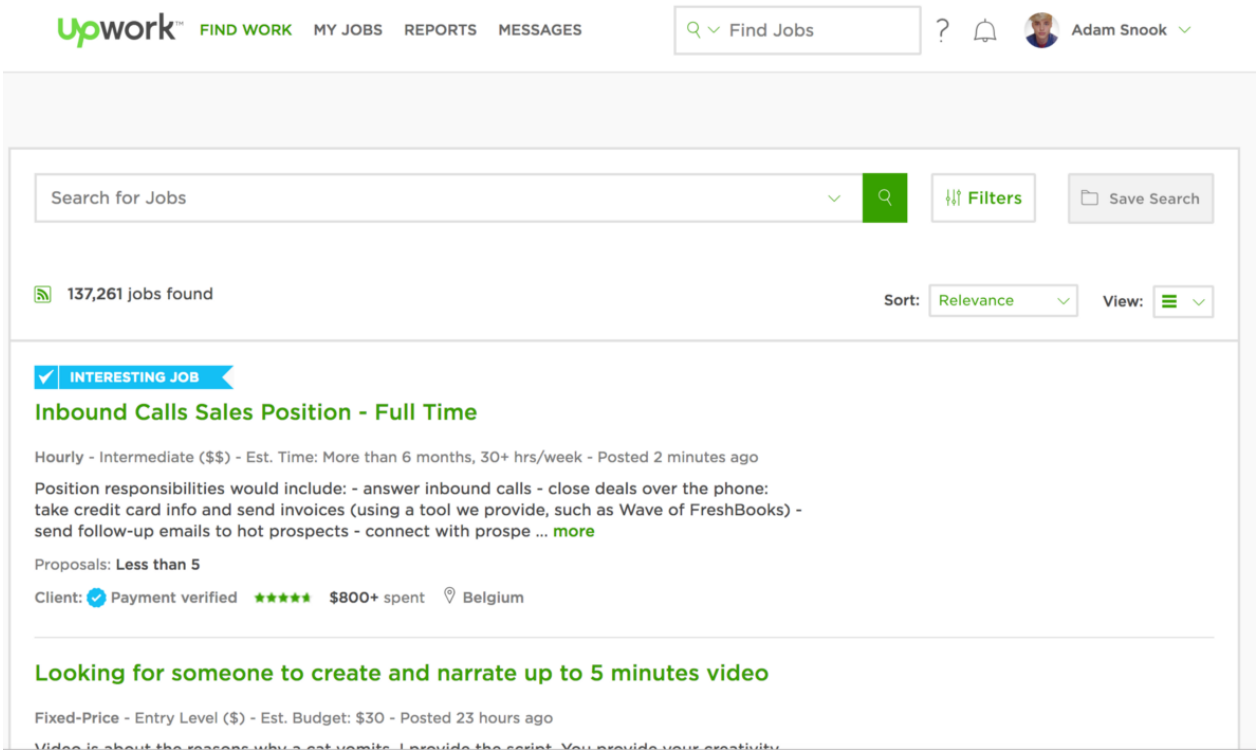


Рисунок 1.4 – Графічний інтерфейс UpWork

Недоліком системи порівняно з платформою для розроблення програмного забезпечення позаштатними працівниками є те що UpWork призначений для пошуку будь-яких спеціалістів, а не лише спеціалістів в галузі розроблення програмного забезпечення і відповідно в ньому немає можливості отримати інформацію потрібну для ознайомлення з процесами розроблення ПЗ, чи інформацію про ефективне управління проектами розроблення ПЗ, адже управління проектами розроблення ПЗ не простий процес, не можна просто додати 10 нових розробників щоб робота йшла швидше, як це можна зробити в інших галузях.

1.4 Веб-сервіс для спільної розробки програмного забезпечення GitHub

Github це здавалося б лише сервіс для збереження вихідних кодів програмного забезпечення, та спільного доступу до них, але цей сервіс містить свою власну підсистему управління проектами під назвою GitHub issues (рисунок 1.5), яка дозволяє створювати задачі, обговорювати їх.

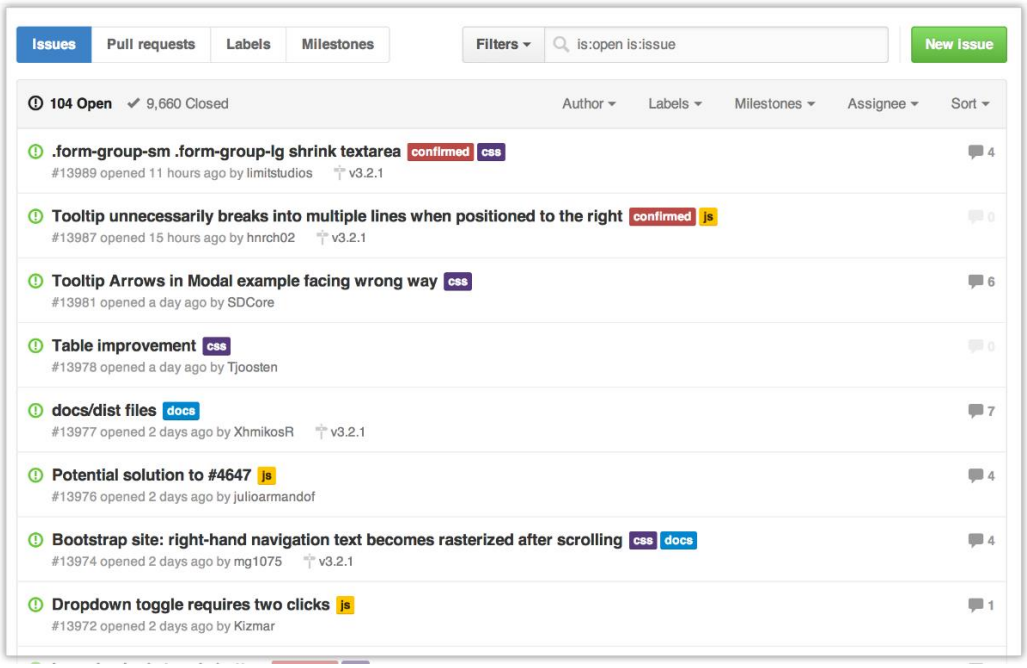


Рисунок 1.5 – Github issues

переглядати рішення до заданих задач через інтегрований порівнювач коду (рисунок 1.6).

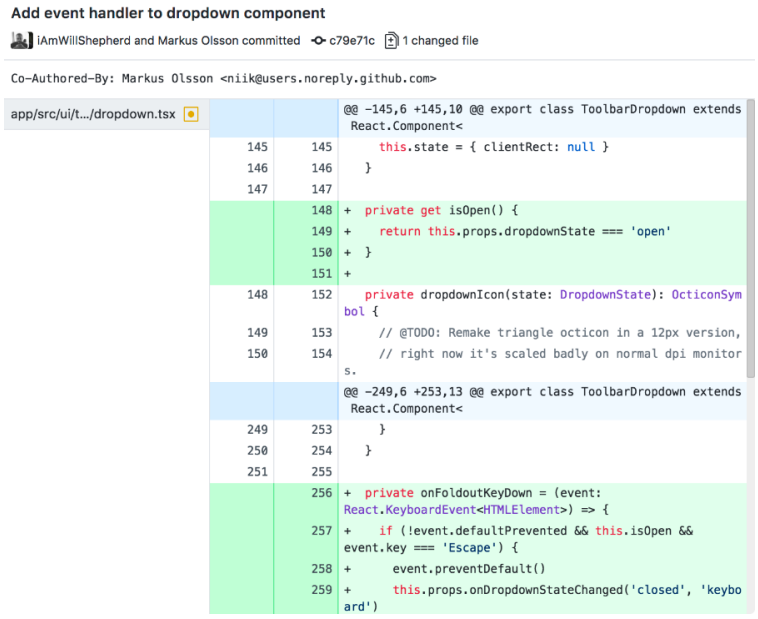


Рисунок 1.6 – Порівнювач коду Github

Порівняння існуючих рішень.

На основі порівняння існуючих рішень, можна сказати що всі рішення значно відрізняються між собою, але всі вони так чи інакше мають застосування в розробленні ПЗ. Розробляти ПЗ використовуючи лише один рішення буде недостатньо і незручно, використовувати всі одразу для різних потреб можна, як це й робиться у більшості випадків при розробленні ПЗ у великих компаніях, буде недосить ефективним через необхідність використання одразу багатьох програмних продуктів та міграції даних між ними і зазвичай дублювання даних в різних рішеннях які не синхронізуються між собою. Але одне, єдине, комплексне рішення яке б об’єднувало весь функціонал оглянутих рішень що стосується розроблення ПЗ, було б дуже доречним, воно і розробляється як платформа для розроблення програмного забезпечення використовуючи ресурси позаштатних працівників, а в подальших планах і штатних працівників.

2 ХАРАКТЕРИСТИКА ПЛАТФОРМИ ДЛЯ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОЗАШТАТНИМИ ПРАЦІВНИКАМИ

2.1 Опис та характеристика роботи платформи для розроблення програмного забезпечення позаштатними працівниками

Завдяки планомірному розвитку та досягненням технічного прогресу багато промислових галузей переходять від продуктових політик до сервісних. Прикладом такої ситуації є купівля телефону будь-якого виробника та вибір компанії, що буде надавати послугу зв’язку (сервіс) та подальша купівля додаткових послуг у цієї компанії за необхідністю в залежності від власних потреб та фінансових можливостей замість покупки супутникового телефону у певної компанії за договором зі сплатою щомісячної абонплати (продуктовий підхід)[19]. Ще одним прикладом описаного підходу є встановлення лічильника використання електроенергії і сплата за сервіс надання доступу до електричних мереж міста згідно обсягів використаної електроенергії та тарифу замість купівлі електрогенератора. Слід зазначити, що подібних прикладів існує багато.

Незважаючи на те, що сфера інформаційних технологій розвивається найбільш динамічно та зазвичай є форпостом запровадження різноманітних нововведень та новаторських рішень, з переходом від продуктового підходу до сервісного існують певні складнощі. Так в частині апаратного забезпечення давно вже практикується надання сервісів оренди серверних потужностей, сервісів хмарних сховищ та хмарних обчислень, сервісів послуг інтернет-провайдерів, тощо. А стосовно програмного забезпечення подібних пропозицій не спостерігається. Існує можливість лише купити програмний продукт, або замовити його розробку, що по суті є аналогічними діями. Слід зазначити, що деякі великі компанії розробники ПЗ, наприклад Microsoft, намагаються запровадити ідею надання послуг використання ПЗ (наприклад, Office 365) [19-26], але такі виключення поодинокі що свідчить про виконання зазначеної означеної тенденції.

Зважаючи на те, що на даний час не існує типових сервісів програмного забезпечення актуальність набуває задача дослідження, якими мають бути такі сервіси і яким вимогам вони мають задовольняти. Отож, основним видом сервісу програмного забезпечення (СПЗ) має стати сервіс безперервної розробки ПЗ (що має замінити собою весь життєвий цикл ПЗ: планування, розробку, тестування, модифікацію, супровід, оплату, тощо). Незважаючи на те, що певні великі компанії-розробники наголошують про надання сервісів ПЗ, насправді вони надають лише можливість вибору однієї з продуктових політик використання вже існуючого продукту.

З врахуванням зазначених обставин виникає необхідність дослідження принципів та умов типової взаємодії замовника послуги ПЗ (клієнта сервісу) та розробника ПЗ (в даному контексті провайдера сервісу). З метою проведення даного дослідження розглянемо основні точки зору та визначимо головних учасників процесу [14-23].

Точка зору клієнта.

Клієнт проходить реєстрацію на сайті сервісу ПЗ як Замовник. Клієнт описує функціонал, який планує отримувати від цільового ПЗ, орієнтовні терміни виходу на робочий режим ПЗ, допустиму вартість умовної години розробки та інші необов’язкові характеристики сервісу. Клієнтську заявку зі статусом «Пошук виконавця» бачать всі зареєстровані розробники ПЗ (провайдери). Зацікавлені провайдери пропонують свої послуги (сервіс ПЗ). Обговорення деталей ведеться на сайті. У клієнта є можливість побачити контактні дані провайдерів та зв’язатися з ними безпосередньо для обговорення деталей та нюансів взаємодії. Після вивчення пропозицій клієнт обирає провайдера та замовляє його послугу ПЗ, заключаючи з ним Договір та закриваючи цим свою заявку. З цього моменту клієнту надається сервіс ПЗ. Клієнт сплачує обумовлену грошову суму на рахунок провайдера, якщо це передбачено. Далі в клієнт створює початкові вимоги до ПЗ. Провайдер оцінює вимоги в умовних трудовитратах. Клієнт має змогу деталізувати вимоги, змінити, відмінити, перевпорядкувати, тощо та запропонувати свою вартість реалізації кожної з вимог. Після змін у вимогах клієнт і провайдер можуть змінити власну

запропоновану ціну реалізації вимог. Після певного ітеративного процесу узгодження вимог та їх вартості (так званий аукціон вимог) клієнт підтверджує готовність обраних вимог до реалізації. Провайдер підтверджує факт запуску вимог у розробку. Після реалізації вимоги до ПЗ провайдер повідомляє клієнта про реалізацію вимоги в новій версії ПЗ, яка автоматично розгортається у клієнта підсистемою розгортання (згідно налаштувань розгортання та політик підтримки версійності ПЗ). Клієнт підтверджує виконання вимоги в повному обсязі після перевірки функціонування ПЗ. В цей момент з рахунку клієнта списується оговорена грошова сума а вимога вважається закритою. В процесі взаємодії клієнта з провайдером можуть виникати нові вимоги, вимоги можуть ієрархічно розбиватися на дрібніші, об'єднуватися, відмінятися і т.д. згідно алгоритму життєвого циклу вимог. Одними з підвидів вимог є вимоги на виправлення помилок, покращення або зміну функціоналу, документування, створення навчальних матеріалів та інші. Надання сервісу провайдером припиняється за двосторонньою згодою, або в випадках передбачених укладеним Договором. Типовим варіантом є нескінченне надання послуги. Переваги, які отримує клієнт від використання концепції програмного сервісу полягають в наступному [14-23]:

− можливість отримання доступу до СПЗ засобами різних обчислювальних пристроїв, таких як ноутбуки, смартфони тощо;

− наявність механізму пошуку та конкурсного відбору провайдера сервісу ПЗ: розробників, тестувальників, дизайнерів тощо;

− наявність середовища для спілкування з провайдером стосовно всіх аспектів замовленого сервісу ПЗ;

− суттєва мінімізація часових ресурсів та індивідуальних зусиль на створення технічного завдання на програмний продукт та розробку ПЗ, а також методик проведення перевірок та операцій тестування разом з іншою супутньою документацією;

− наявність можливості формування поточних версій деяких паперових та електронних документів в будь-який час, таких як ТЗ, звіти, тощо;

− цілковита прозорість виконаних робіт провайдером згідно поставленим

вимогам;

− можливість надгнучкої зміни вимог до ПЗ на всіх етапах розробки та супроводу;

− наявність системи встановлення пріоритетів до реалізації вказаних вимог, узгоджена з провайдером;

− наявність середовища врахування помилок та відстеження, виправлення та попередження неузгодженостей;

− володіння можливостями вільного доступу до сховища поточних напрацювань, до яких належать вихідні коди, скомпільовані модулі, документація, тощо;

− можливість залучення професійного ревізора для оцінення ступеня реалізації поставлених вимог до ПЗ;

− своєчасна можливість повного або часткового залучення зацікавлених осіб в процес розробки зі сторони замовника згідно обраної рольової політики в межах надання сервісу ПЗ;

− суттєве спрощення процесу розгортання та оновлення ПЗ;

− з точки зору фінансової частини своєчасність та оперативність оплати робіт виконаних провайдером;

− можливість повного відстеження та планування бюджету на сервіс ПЗ;

− гнучка можливість зміни провайдера в будь-який час без втрати існуючих напрацювань та історії змін вимог до ПЗ;

Точка зору провайдера.

Провайдер проходить реєстрацію на сайті сервісу ПЗ як Провайдер. Провайдер вказує профільні предметні області (в яких має певні напрацювання та/або конкурентні переваги), посилання на інформацію по реалізованим проектам, кваліфікацію та склад команди розробки та супроводження ПЗ, орієнтовну вартість години надання СПЗ, тощо. Профіль Провайдера разом з рейтингом, відгуками та іншими оцінками бачать як всі зареєстровані клієнти СПЗ так і інші розробники ПЗ (провайдери). Зацікавлені клієнти звертаються до Провайдера з метою уточнення деталей надання СПЗ для можливої подальшої співпраці. Інші провайдери мають

змогу звернутися до іншого Провайдер як до співвиконавця з метою доручити Провайдеру виконання певних робіт в межах їх власного СПЗ (тобто реалізується платформа пошуку виконавців робіт, але з урахуванням їх планової зайнятості та інших факторів). В Провайдера є можливість побачити контактні дані клієнтів (якщо вони не вказали зворотного) та зв'язатися з ними безпосередньо для обговорення пропозиції по цільовому СПЗ. Після обговорення пропозицій Провайдер вкладає договір на надання СПЗ з клієнтами або іншими провайдерами. Провайдер розпочинає уточнення початкових вимог замовника до ПЗ за допомогою механізму «аукціон задач» на базовому рівні (тобто до рівня достатнього для планування початкових робіт). Після підтвердження клієнтом Провайдер починає виконання робіт по реалізації вказаних вимог. Після виконання вимоги Провайдер очікує оцінки клієнта на відповідність реалізації. В разі підтвердження клієнтом повного виконання вимоги проводиться фіксація в системі інформації про елемент оплати (фізичне перерахування коштів з таким обліком напряму не зв'язане). Подальші вимоги до ПЗ з'являються в системі в процесі надання СПЗ. Вони можуть генеруватися замовником або розробником з обов'язковим затвердженням зацікавленими сторонами. Переваги, які Провайдер отримує від використання концепції програмного сервісу полягають в наступному [23]:

− можливість отримання доступу до сервісу ПЗ засобами різних обчислювальних пристроїв, таких як ноутбуки, смартфони тощо;

− наявність механізму пошуку та конкурсного відбору провайдера сервісу ПЗ: розробників, тестувальників, дизайнерів тощо;

− наявність середовища для спілкування з провайдером стосовно всіх аспектів замовленого сервісу ПЗ;

− суттєва мінімізація часових ресурсів та індивідуальних зусиль на створення технічного завдання на програмний продукт та розробку ПЗ, а також методик проведення перевірок та операцій тестування разом з іншою супутньою документацією;

− наявність можливості формування поточних версій деяких паперових та електронних документів в будь-який час, таких як ТЗ, звіти, тощо;

− можливість надгнучкої зміни вимог до ПЗ на всіх етапах розробки та супроводу;

− наявність системи встановлення пріоритетів до реалізації вказаних вимог, узгоджена з провайдером;

− наявність середовища врахування помилок та відстеження, виправлення та попередження неузгодженостей;

− своєчасно надана можливість залучення до виконання робіт з розробки програмного забезпечення інших провайдерів СПЗ в якості підрядників;

− спрощення розгортання та оновлення ПЗ;

− можливість передачі клієнта іншому провайдеру (за згоди клієнта) без втрати існуючих напрацювань та історії змін вимог до ПЗ;

− суттєве спрощення процесу розгортання та оновлення ПЗ;

− з точки зору фінансової частини своєчасність та оперативність оплати робіт виконаних провайдером;

− стабільне гарантування оплати виконаних робіт клієнтом незалежно від факту повної реалізації вимог у випадку зміни вимог клієнтом, або при припиненні робіт з ініціативи клієнта;

− наявність гнучких механізмів прогнозування та відстеження часових термінів реалізації вимог, задач, робіт, тощо;

− повна мінімізація бюрократичних витрат;

− значне підвищення взаєморозуміння з клієнтом та максимальне уникнення конфліктних ситуацій;

− коректна реалізація процедур та механізмів можливого вирішення конфліктів між учасниками процесу за допомогою залучення третьої сторони;

− наявність середовища тісної інтеграції зовнішніх систем розробки та супроводження ПЗ клієнтів в спеціалізовані АРМ провайдера;

Платформа розроблення програмного забезпечення, повинна бути спроектована так, щоб її можна було інтегрувати в іншу систему, для цього платформа повинна мати документований інтерфейс програмування застосунків [14-23], що дозволить використовувати її в інших підсистемах програмного сервісу

таких як аукціон вимог, система керування проектами, система контролю версій та інших. Після інтеграції з іншими підсистемами програмного сервісу, клієнти платформи зможуть створювати власні робочі процеси розроблення програмного забезпечення та матимуть змогу більш гнучко керувати розробленням своїх проектів, використовуючи різні підсистеми програмного сервісу, що в результаті дозволить:

- швидко змінювати вимоги;
- можливість відстеження та трекінгу помилок;
- створювати за змінювати технічне завдання користуючись спеціальними шаблонами;
- використовувати внутрішні хмарні сервіси інтегровані в платформу;
- створення документації з використанням шаблонів;
- використання внутрішньо сервісних програмних бібліотек;
- використання шаблонів створення типових програмних застосунків (інтернет магазин, система обліку, інформаційний веб-сайт…);
- оплата послуг використовуючи платіжну підсистему програмного сервісу.

Одним з етапів реалізації програмного сервісу є створення платформи для розроблення програмного забезпечення використовуючи ресурси позаштатних працівників, що в свою чергу потребує адаптації до «сервісу безперервної розробки ПЗ» реалізувавши власний підхід до життєвого циклу програмного забезпечення спрямований на вирішення проблем що стосуються позаштатності працівників. А саме позаштатний працівник може виконувати наступні дії, що можуть призвести до певних проблем:

- припинити свою діяльність в будь-який, зручний для нього момент;
- створити працююче програмне забезпечення, що має суттєві недоліки;
- писати незрозумілий іншим розробникам програмний код.

Вирішити кожну з проблем можна декількома варіантами, які може обрати клієнт:

- для вирішення даної проблеми можна, укласти угоду з працівником про

виконання поставлених задач, але це не завжди можливо.

− використовувати загальнодоступну рейтингову систему, яку бачать всі можливі клієнти, для оцінки працівників;

− оплачувати лише повністю виконане завдання;

− протестувати виконане завдання;

− наймати досвідчених працівників;

− використовуючи рейтингову систему публікувати знайдені «незрозумілості»;

− проводити огляд коду користуючись послугами досвідчених розробників.

Підсумовуючи сказане можна виділити етапи робочого процесу розроблення програмного забезпечення використовуючи ресурси позаштатних працівників, наведені на рисунку 2.1, використання представлених на ньому етапів розроблення, дозволяє отримати на виході якісне програмне забезпечення, незважаючи на позаштатність працівників, оскільки зазвичай при розробленні ПЗ позаштатні працівники отримують вимоги від замовника, розробляють ПЗ, віддають замовнику, а замовник отримує зовсім не той продукт на який очікував. В представленому робочому процесі немає нічого незвичайного для великих корпорацій, що займаються розробленням ПЗ, але такий робочий процес дуже незвичний для позаштатних працівників, оскільки вони не звикли працювати в команді і зазвичай розробник є сам собі бізнес-аналітиком, тестувальником, DevOps інженером.

Всі вищесказані фактори дуже сильно впливають на якість розробленого ПЗ, причому на всі характеристики якості, такі як:

− зручність супроводження;

− надійність;

− продуктивність;

− практичність;

− функціональність;

− здатності портування ПЗ на іншу програмну платформу.

Таким чином, використання наведеного далі робочого процесу суттєво покращує якість розробленого позаштатними працівниками ПЗ порівняно з

розробленням однією людиною, при замовленні розроблення ПЗ на веб-ресурсах пошуку позаштатних працівників, оскільки замовник отримає ще додаткову інформацію для супроводження ПЗ використовуючи ресурси працівників, що попереднь не були виконавцями його замовлення.

Рисунок 2.1 – Етапи робочого процесу розроблення програмного забезпечення.

Всі етапи процесу можна скомбінувати в єдиний «Сервіс безперервної розробки ПЗ» та надавати його як рішення для тих, хто мало розуміється на створенні програмного забезпечення, але хоче отримати на виході якісний продукт.

Більш детально цей підхід виглядатиме наступним чином. Клієнт розміщує в сервісі замовлення і починається перший етап, клієнту пропонується обрати людину яка допоможе сформувати вимоги, перший пункт вимог полягає в створенні макета бажаного програмного забезпечення, для того, щоб клієнт зміг переконатися в тому, що реалізований програмний продукт задовольняє його вимогам вчасно, а не під час кінцевого етапу розроблення програмного забезпечення.

Після формування вимог, клієнт переходить на наступний етап де йому пропонуються працівники які можуть реалізувати проект, запропоновані працівники формуються з вимог поставлених на попередньому етапі, наприклад у вимогах було вказано що проект має реалізовуватися на Java і клієнт отримає відповідну вибірку розробників, що володіють даною технологією. Далі клієнт обирає необхідних йому розробників (на основі вимог першого етапу) та чекає на першу реалізацію.

Після отримання реалізованого програмного продукту, його програмний код передається на огляд досвідченим працівникам (код перевіряється на зрозумілість читабельність, правильність з точки зору реалізації різних підходів до архітектури програмного забезпечення), на основі огляду вносяться правки, та проводиться повторний огляд.

Після правок клієнт отримує програмне забезпечення яке зможуть дороблювати інші розробники. На наступному етапі клієнту буде запропоновано обрати тестувальника, запропоновані клієнту тестувальники формуються з вимог поставлених на етапі формування вимог, наприклад у вимогах було вказано що проект має реалізовуватися у вигляді REST API і клієнт отримає відповідну вибірку тестувальників що володіють даною технологією.

Далі клієнту буде запропоновано розгорнути створене програмне забезпечення в хмарному сервісі, і відповідний персонал, який зможе це зробити, або клієнт отримає продукт у вигляді програмного забезпечення та вихідного коду.

Перед кожним етапом проводиться так званий «аукціон задач»[19] що дозволяє узгодити ціну на послугу того чи іншого працівника. Наприклад для формування вимог це працює так: клієнт розміщує замовлення на формування вимог і йому надходять пропозиції людей, що можуть сформувати вимоги, і відповідно ціни на їх послуги, клієнт може знайти виконавця послуги і погодитися з його ціною, або запропонувати свою ціну.

Платформа для розроблення програмного забезпечення дозволяє її користувачам отримати доступ до різних технологій програмування, що дозволяють пришвидшити і полегшити їх роботу завдяки вже налагодженим процесам. Оскільки платформа розробляється для використання ресурсів позаштатних працівників, технології програмування будуть спрямовані саме на позаштатність.

Для того щоб зрозуміти, що таке технологія програмування взагалі, і що таке гнучкі технології програмування зокрема, необхідно ознайомитися зі специфікою проблем, які виникають при розробленні сучасного програмного забезпечення. Грубо кажучи, при розробленні будь-якого програмного продукту необхідно повністю і в встановлені терміни вирішити поставлене завдання, забезпечивши при цьому високий рівень якості. Дані вимоги можна пред'явити як до виконання лабораторної роботи по програмування, так і до розробки сучасного програмного комплексу. Однак в останньому випадку розробник зіткнеться з вирішенням завдань, які, на перший погляд, не відносяться до дисципліни програмування (у вузькому сенсі). такими завданнями є, наприклад, визначення та специфікація вимог, розробка архітектури системи, управління якістю продукту і так далі. Саме для вирішення подібних завдань і необхідна технологія програмування. При цьому важливо відзначити, що технологія програмування не є набором абстрактним умовиводів і формальних методик. Технологія програмування пронизує всі фази життєвого циклу програмного продукту. Будь-яка дія, що здійснюється в процесі розробки продукту, виконується відповідно до використовуваної технологією програмування. Також важливо відзначити відмінність між методологією і технологією програмування.

Методологія програмування є формою приписів і норм, в яких фіксуються

зміст і послідовність певних видів діяльності. технологія програмування є сукупністю прийомів і способів виконання певних видів діяльності. Виходячи з вищенаведених визначень ясно, що технологія програмування присутня при розробці будь-якого програмного продукту, навіть якщо це і не закріплено формальної методологією розробки. Одним з ключових понять технології розроблення програмного забезпечення, як і багатьох інших областей діяльності, є поняття проекту. Проект є унікальне тимчасове підприємство, спрямоване на створення певного, унікального продукту і послуги. технологія управління проектом є сукупність знань, навичок, інструментів і методів для планування і реалізації дій, спрямованих на досягнення поставленої в рамках проекту мети. Сучасні програмні системи розробляються в виключно складній обстановці. На шляху сучасного проекту по розробленню програмного забезпечення постають численні організаційні та технічні перешкоди. Можна виділити наступні змінні, які впливають на складність проекту по розробці програмного забезпечення:

− наявність висококваліфікованих фахівців на ринку праці. Причому чим новіше використовувана технологія, тим менше доступно фахівців, їй володіють;

− стабільність використовуваної технологічної платформи. чим новіше використовувана платформа, тим менше її стабільність;

− стабільність і функціональність використовуваних інструментів розробки. Чим новіший і могутніше використовуваний інструмент розробки, тим менше фахівців здатні з ним ефективно працювати, і тим менш стабільна його функціональність;

− ефективність використовуваних методів розробки, включаючи методи моделювання, проектування, тестування і управління версіями;

− доступність фахівців, що володіють експертизою в прикладної області;

− використана методологія і її відповідність даним проектом;

− ситуація на ринку і її вплив на терміни проекту і плановану функціональність продукту;

− терміни і фінансування проекту та безліч інших організаційних і технічних

змінних.

Можна припустити, що складність проекту є функцією від вищенаведених змінних. Важливо розуміти, що будь-яка з цих змінних може змінювати своє значення протягом життєвого циклу проекту. Наприклад, може істотно змінитися ситуація на ринку, що може привести до стиснення термінів і збільшення обсягу запланованих робіт. Для управління проектами з високою складністю потрібні розвинені інструменти контролю і управління ризиками. Однак в управлінні проектами виділені наступні проблеми.

Більшість процесів розроблення некеровані. вихідні дані і бажаний результат багатьох процесів розробки невідомі, або визначені дуже нечітко. Більш того, процес досягнення бажаного результату не піддається формалізації. Прикладами некерованих процесів розробки є розробка архітектури та вичерпне тестування продукту. Ідентифіковані процеси розробки супроводжуються невідомою кількістю неідентифікованих процесів розробки. Наприклад, в процесі моделювання і докази адекватності моделі виникає безліч процесів розробки, які практично не піддаються ідентифікації та формалізації.

Вимоги до продукту часто змінюються протягом життєвого циклу проекту, що вимагає складної процедури зміни і узгодження вимог. Спроби запропонувати формальну, деталізовану методологію розробки програмного забезпечення виявляються безуспішними, тому що сам процес розробки не піддається деталізації і формалізації. Сліпе слідування методологій, що передбачає керованість і передбачуваність процесів розробки призводить до непередбачуваних результатами проекту.

Один з основних принципів гнучких технологій розробки програмного забезпечення – відмова від тривалого проектування перед початком роботи і виконання проектування протягом усього виконання проекту. Методики, які передбачали виконувати всі проектування до початку роботи над проектом, засновані на помилковому уявленні про те, що заздалегідь можна визначити всі вимоги до програмного продукту, розробити план і виконати проектування. У гнучких технологіях розробки програмного забезпечення передбачається, що на

початку роботи розробники мають у своєму розпорядженні тільки приблизний план реалізації, який постійно разом з програмним продуктом розвивається і уточнюється в процесі роботи. У гнучких технологіях розробки ПЗ етап проектування прийнято називати дизайном. Противники гнучких технологій стверджують, що дизайн в гнучких технологіях і проектування в класичних методиках розробки програмного забезпечення – абсолютно різні поняття, але це неправильне твердження. Гнучкі технології припускають безперервне здійснення процесу проектування, а не тільки на самому початку роботи над проектом.

На початку проекту виконується лише невелика частина роботи – формування загального уявлення. Для цієї мети в гнучких технологіях використовуються системні метафори, на основі яких формується високорівнева схема проекту. Еволюційний підхід спочатку лежав в основі проектування програмного забезпечення, але в умовах неорганізованості і неструктурованості процесу розробки від нього відмовилися на користь попереднього проектування. При цьому в якості прототипів процесів проектування в класичних технологіях розробки програмних продуктів приймалися методи проектування, які добре зарекомендували себе при проектуванні складних будівельних або промислових об'єктів. Такий підхід виявляється абсолютно неприйнятним при створенні програмних продуктів, вимоги до яких перед початком роботи остаточно не сформовані. Робочий процес що призначений для позаштатних працівників наведений в додатку Рисунок 2.1.

Якщо в умовах неповної визначеності (замовник не до кінця усвідомив вимоги або розробники неправильно інтерпретували його побажання) спробувати виконати проектування, а потім відповідно до розробленого проекту реалізувати програмний продукт, то отриманий результат, як правило, не зможе задовольнити замовника. Рішення, що дозволяє вийти із ситуації, може варіюватися від нескінченної низки безладних змін в проекті (це повертає розробників до первісного, безладного способу проектування) до повного перепроектування на основі нових вимог, при цьому вартість внесення змін до проекту експоненційно зростає.

Отже узагальнюючи все вищесказане можна зробити висновок, що

розробляєма платформа поєднує в собі різні системи для розробки ПЗ такі як: систему для пошуку людей необхідних для розробки, супроводу, консультації, тестування, розгортання проекту, систему для управління проектом, систему для розгортання проекту, систему для перегляду і ведення документації, систему для оплати послуг також систему з уже готових програмних бібліотек для розроблення різних типів програмних додатків. І в цілому використовуючи дану платформу для розроблення можна буде використовувати лише її та будь-яку сумісну з нею IDE, а всі необхідні функції з раніше різних систем будуть в одній єдиній платформі.

2.2 Вимоги до характеристик розробляємої платформи

Підставою для розробки технічного завдання є завдання з дипломного проектування.

Платформа розроблення програмного забезпечення позаштатними працівниками призначена для оптимізації процесів пошуку виконавців робіт з реалізації програмного забезпечення на замовлення, а також супроводу всіх етапів розробки програмного забезпечення.

Експлуатація розробленої платформи передбачається шляхом звернення до бази даних з клієнтського веб-додатку, користувачами яких є клієнти та провайдери платформи.

Застосування. Платформа розроблення програмного забезпечення позаштатними працівниками дозволяє підвищити швидкість і якість процесів розробки програмного забезпечення.

Обов'язкові вимоги:

- можливість запису даних про клієнтів та провайдерів в базу;
- реєстрація нових заяв на розробку;
- надання можливостей імпорту даних з систем керування проектами, та систем контролю версій;
- контроль поточного стану виконання замовлень;

– контроль фінансових операцій з оплати послуг;

– реєстрація та контроль зміни вимог до програмного забезпечення;

– створення документації та звітів за різними показниками;

– передача оперативних і аналітичних звітів в інші програмні засоби з метою їх подальшої обробки за допомогою спеціалізованих функцій інших програмних засобів.

Вимоги до способів і засобів зв'язку для інформаційного обміну між компонентами системи:

Інформаційний обмін здійснюється за допомогою архітектури «клієнт-сервер».

Вимоги по архітектурі − платформа, що розробляється, повинна бути масштабованою та гнучкою, зберігати незалежність від джерела даних. Це повинно бути досягнуто за рахунок використання клієнт-серверної архітектури і сучасної мови програмування.

Вимоги до інтерфейсу − програмна частина повинна володіти дружнім, інтуїтивно зрозумілим користувачеві графічним інтерфейсом, що дозволяє здійснювати в повній мірі всі функціональні можливості системи.

Вимоги до апаратних і програмних ресурсів − передбачається робота додатку шляхом звернення до сервера. Засоби, необхідні для роботи програми, не повинні створювати конфліктних ситуацій при організації інформаційних потоків між додатками користувачів і сервером. Запити до сервера повинні бути побудовані таким чином, щоб забезпечувати необхідну швидкодію системи.

Вимоги до програмного забезпечення системи − програмний додаток платформи повинен працювати на базі 32- і 64-разряних операційних систем Windows.

Вимоги до технічного забезпечення системи − в комплекс технічних засобів повинні входити такі елементи:

− автоматизовані робочі станції;

− джерело безперебійного живлення;

− середовище передачі даних між робочими станціями.

Для збору інформації від АРМ необхідна клавіатура, мишка із двома кнопками та колесом та монітор для відображення інформації. У випадку смартфона чи планшета достатньо сенсорного екрану від 3 дюймів діагоналі, модуля 3G або Wi-Fi.

Перед встановленням платформи для розроблення програмного забезпечення варто провести заходи з підвищення фізичної безпеки. Для підвищення фізичної безпеки під час встановлення програмної платформи необхідне виконання наступних дій:

− розташувати веб-сервер та сервер баз даних в захищеному приміщенні;

− ввести записи змін у вихідних кодах скриптів та транзакцій баз даних;

− регулярно створювати резервні копії бази даних і зберігати їх в безпечному місці за межами розташування комп’ютера;

− використовувати дисковий масив (RAID) для найбільш критичних файлів даних.

Варто враховувати при проектуванні призначених для користувача інтерфейсів наступні вимоги:

− видимість стану системи: система повинна завжди і за прийнятний час реагувати на дії користувача і інформувати його про поточний стан роботи;

− відповідність системи та реального світу: система повинна бути зрозумілою користувачеві, використовуючи слова, фрази і концепції, які вже відомі користувачеві. Подання інформації повинно бути організовано згідно природного і логічного порядку;

− свобода виконуваних дій користувача: користувач повинен володіти системою і мати можливість змінити поточний стан програми шляхом скасування або повторення операцій;

− використання послідовності дій і залучення стандартів: принцип послідовності означає використання одних і тих же понять і засобів для відображення схожих образів і виконання однотипних дій. Найлегше це досягається шляхом використання типових для конкретної платформи рекомендацій і угод;

− виправлення та запобігання помилок: система повинна бути розроблена так, щоб мінімізувати число ситуацій, в яких користувач має змогу помилитися. Це краще, ніж випадок, коли з'явиться інформація про виниклу проблему. Як відомо, хвороба легше попередити, ніж лікувати;

− розуміння краще, ніж механічне запам'ятовування: всі об'єкти, функції, дії повинні бути видні користувачеві. Він не повинен запам'ятовувати і утримувати в пам'яті інформацію з однієї частини діалогу, щоб застосувати її в інший. У будь-який момент користувачеві повинно бути ясно, що потрібно робити в даний момент. У разі необхідності, користувач повинен мати простий доступ до контекстної довідки;

− гнучкість і ефективність використання: інтерфейс програми повинен бути однаково зручний як для новачків, так і для досвідчених користувачів, необхідно передбачити можливість забезпечення альтернативних способів роботи з ним. Таких як «гарячі» клавіші, тулбари, контекстні меню, щоб користувач сам вибрав те, що йому зручніше;

− естетичний і мінімалістичний дизайн: діалоги не повинні містити нерелевантну або рідко використовувану інформацію. Кожен новий елемент інтерфейсу конкурує з іншими і відволікає частину уваги користувача, тим самим зменшуючи відносну видимість дійсно необхідної інформації;

− розпізнавання і нейтралізація помилок: необхідно допомагати користувачеві розпізнавати, діагностувати і виправляти помилки, тому що повідомлення про помилки повинні бути виражені простою мовою (без кодів), точно описувати проблему і пропонувати конструктивне рішення для неї;

− довідка та документація: краща система та, яка може бути використана без будь-якої документації. Це ідеал, але в реальності програма повинна містити необхідну довідкову інформацію та документацію. Будь-яка (довідкова) інформація повинна бути доступна для пошуку, сфокусована на задачах користувача, послідовна в описі його дій і, при цьому, повинна бути не надто громіздкою.

При проектуванні моделі платформи для розроблення програмного забезпечення позаштатними працівниками необхідно враховувати специфіку

позаштатності працівників, що дозволить обмежити можливості використання системи і відповідно полегшити її реалізацію, а у випадку успішності впровадження платформи можна буде додати функціонал для використання платформи штатними працівниками, а також подальшу можливість реалізації розглянутої платформи на основі інтеграції її з популярними зовнішніми системами контролю версій, системами управління проектами, системами пошуку працівників, для того щоб нові користувачі могли імпортувати дані з систем, якими вони користуються і перейти на використання платформи для розроблення ПЗ позаштатними працівниками. Однією з головних вимог до розробляємої системи є вирішення базового загальновідомого робочого процесу з розроблення ПЗ з використанням різноманітних підсистем, таких як: система управління проектами, система контролю версій, система розгортання ПЗ в хмарних сервісах, система комунікації, система пошуку працівників, система оплати послуг. У випадку якщо запропонований робочий процес не влаштовуватиме замовників, то його можна легко змінити і скоріше за все з часом він буде вдосконалюватися, щоб зберегти баланс між швидкістю розроблення, якістю та ціною розроблення ПЗ, а от якщо взаємодія з однією з вищевказаних систем працюватиме некоректно то користувачі платформи відмовляться від її використання на користь перевірених часом набором улюблених систем, якими вони звикли користуватися.

3 ПРОЕКТУВАННЯ ПЛАТФОРМИ ДЛЯ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОЗАШТАТНИМИ ПРАЦІВНИКАМИ

3.1 Обґрунтування вибору підходів і технологій

При реалізації платформи для розроблення програмного забезпечення позаштатними працівниками передбачається використання бази даних, функціонуючої на основі СУБД MS SQL Server.

Одна з найбільших переваг реляційної моделі полягає в тому, що цілісність даних – невід'ємна частина моделі. Цілісність даних, реалізована як компонент моделі, а саме як компонент визначень таблиці вважається декларативною цілісністю даних.

При проектуванні моделі даних для своєї бази даних, в процесі розробки слід ретельно враховувати всі суб'єктивні потреби та додатки для того, щоб задати адекватні визначення всіх вхідних об'єктів.

MS SQL Server дозволяє використовувати декілька його екземплярів одночасно щоб підвищити продуктивність та надійність системи, що неодмінно потрібно для розробляємої платформи, оскільки в неї буде дуже велике навантаження з боку клієнтів, також не слід забувати про надійність системи, завдяки резервному копіювання одного екземпляру на інший можна бути впевненим, що у випадку неполадок з одним екземпляром, платформа використовуватиме його резервну копію з іншого екземпляру.

Екземпляр SQL Server – це установка механізму служби бази даних SQL Server. На одному комп'ютері можна встановити кілька екземплярів SQL Server. Кожен з них повністю незалежний від інших екземплярів з точки зору безпеки даних, якими він управляє, і у всіх інших відносинах.

На логічному рівні у двох екземплярів, розміщених на одному і тому ж комп'ютері, загального стільки ж, скільки у двох екземплярів, що знаходяться на різних комп'ютерах. Звичайно ж, вони спільно використовують фізичні ресурси

сервера, такі як центральний процесор, оперативна пам'ять і диски.

Один з екземплярів, встановлених на комп'ютері, може бути заданий як екземпляр за замовчуванням, а інші повинні бути іменованими екземплярами. Задається екземпляр як екземпляр за замовчуванням або іменований в процесі установки, але пізніше змінити цей вибір не можна.

3.2 Проектування бази даних

3.2.1 Опис концептуальної моделі даних

Проектування бази даних представляє собою процес, який складається з декількох етапів, послідовність виконання яких забезпечує коректність та ефективність процесів роботи з даними протягом подальшої експлуатації бази даних [1-14]. Структура процесу проектування бази даних за етапами наведена на рис. 3.1.

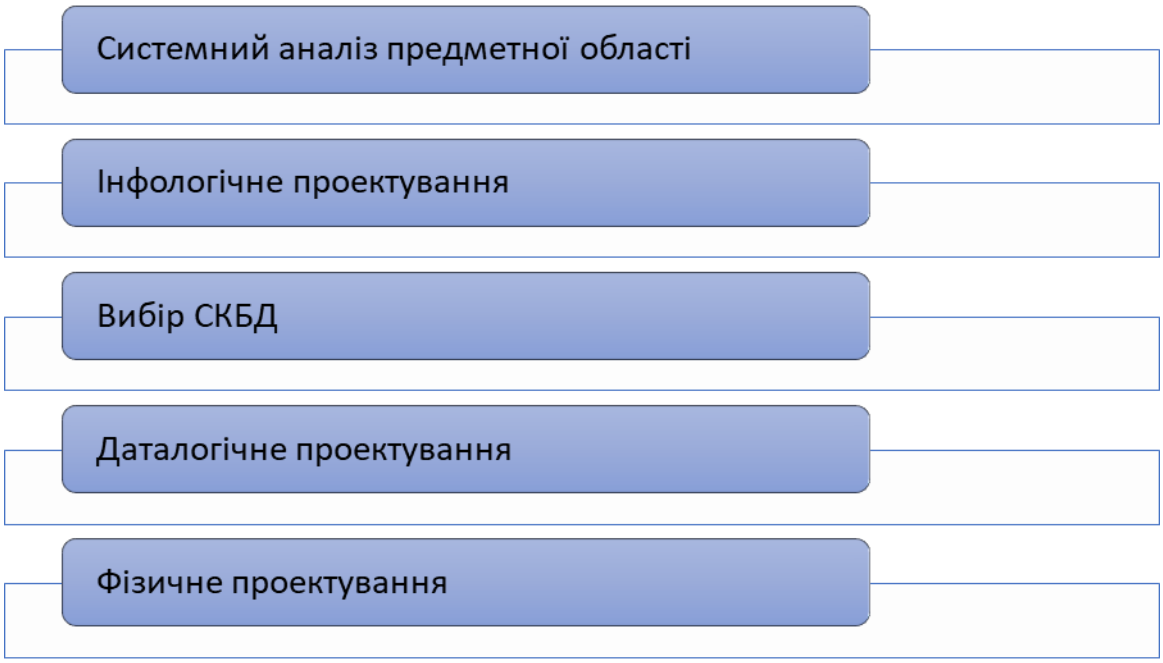


Рисунок 3.1 – Етапи проектування бази даних

Для досліджуваної предметної області побудовано ER-діаграму засобами Erwin Model Navigator представлену на рисунку 3.2. ER-діаграма призначена для аналізу предметної області та початкового етапу проектування БД, в якому

визначаються найголовніші сутності, над якими в подальшому потрібно буде визначити атрибути та зв’язки.

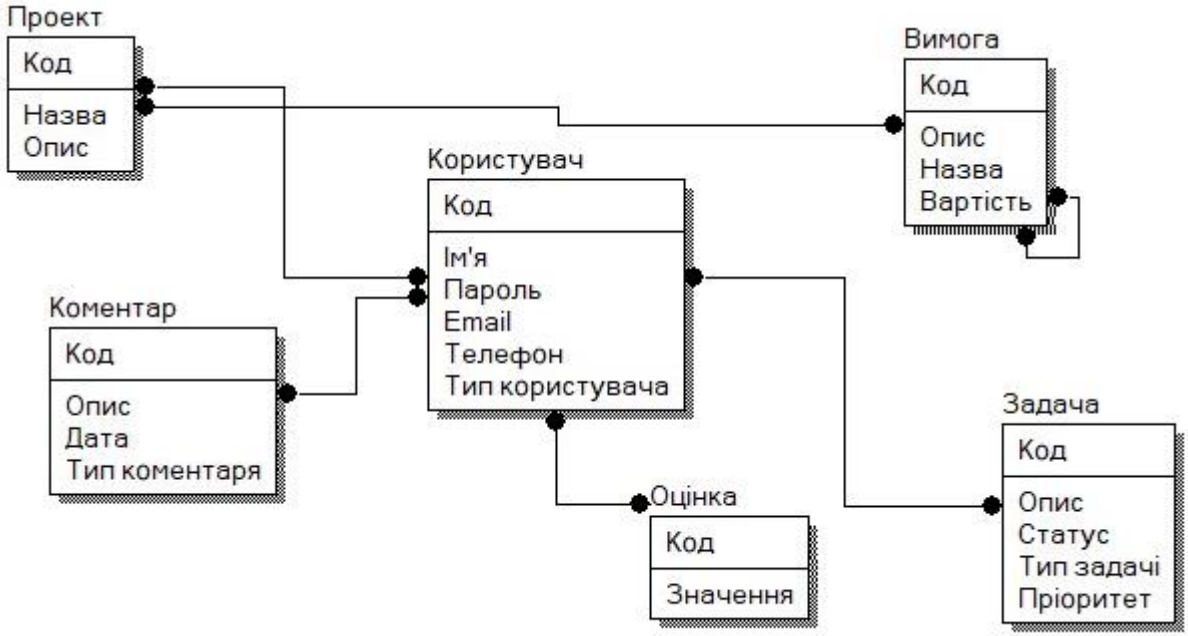


Рисунок 3.2 – ER діаграма

В результаті аналізу предметної області виявлено основні сутності, опис яких необхідно здійснити при інфологічному проектуванні.

Найбільш активною та змістовною з точки зору інформаційного наповнення системи є сутність «Користувач», тому саме з цієї сутності слід почати процес інфологічного проектування.

Проектування бази даних здійснено за допомогою методу «сутність - зв'язок». При проведенні аналізу предметної області виділено такі сутності:

- користувач;
- проект;
- вимога;
- задача;
- оцінка;
- коментар.

Для всіх зазначених сутностей виділені атрибути – властивості, характерні для

всіх екземплярів певної сутності. Для забезпечення унікальності записів в таблицях баз даних екземпляру кожної сутності відповідає унікальний код. На рис. 3.3-3.18 зображено діаграми з позначенням атрибутів відокремлених сутностей.

Наведені на рис. 3.3 атрибут користувача «Тип користувача», який має зберігатися в базі даних платформи розробки програмного забезпечення, в силу її самостійної інформаційної наповненості доцільно представити у вигляді окремої підпорядкованої сутності, що дозволить шляхом використання механізмів роботи зовнішніх ключів досягти цілісності та запобігти суперечності даних.

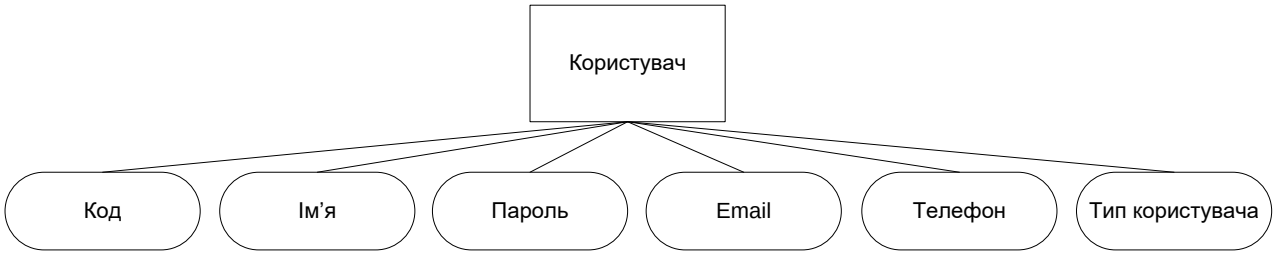


Рисунок 3.3 – Атрибути сутності «Користувач»

Атрибути сутностей «Проект» та «Вимога» (рис. 3.4 – 3.5) представлені описом та назвою. Для вимоги вказується вартість як додатковий атрибут.

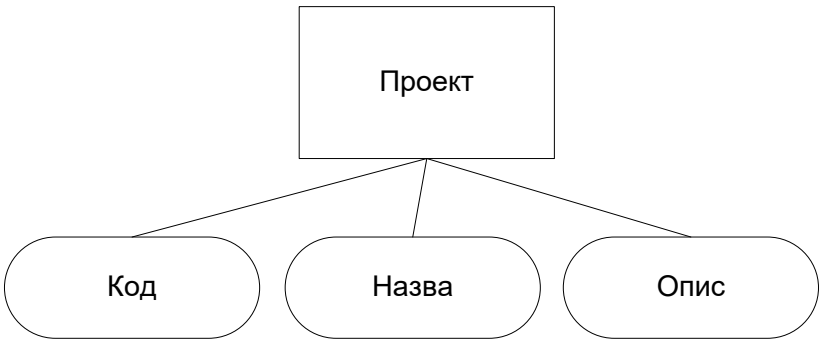


Рисунок 3.4 – Атрибути сутності «Проект»

Проект містить три основних атрибути притаманні новим проектам, містить назву, код та короткий опис проекту. Код та назва призначені для його ідентифікації в платформі, що розробляється, а опис для пояснення головної мети проекту, та коротких характеристик, що стосуються його специфіки, щоб в подальшому

пропонувати користувачам «проекту» різні технології програмування, які можуть бути корисні їм в роботі над проектом.

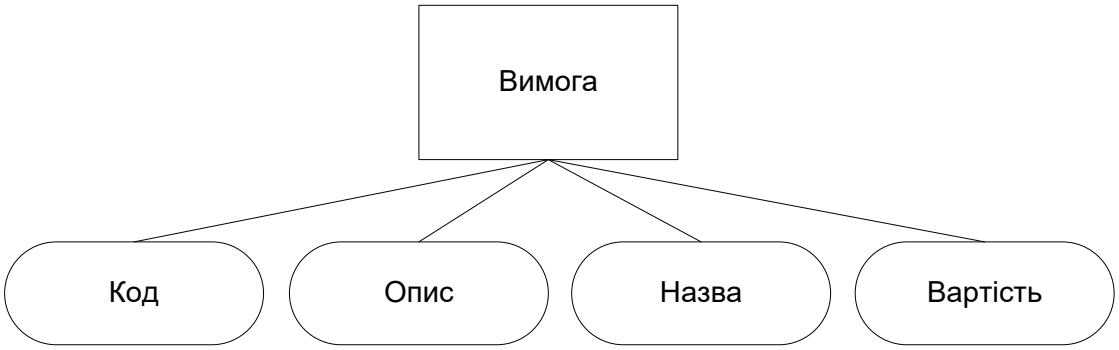


Рисунок 3.5 – Атрибути сутності «Вимога»

Для сутності «Задача» також знайдено атрибути статус, тип задачі та пріоритет, які можуть бути декомпозовані в окрему підпорядковану сутність. Аналогічно з атрибутом «Тип коментаря» (рис. 3.6 – 3.8).

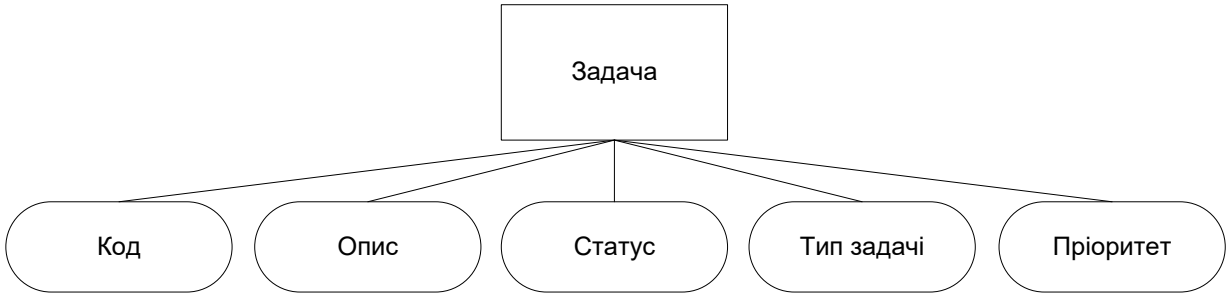


Рисунок 3.6 – Атрибути сутності «Задача»

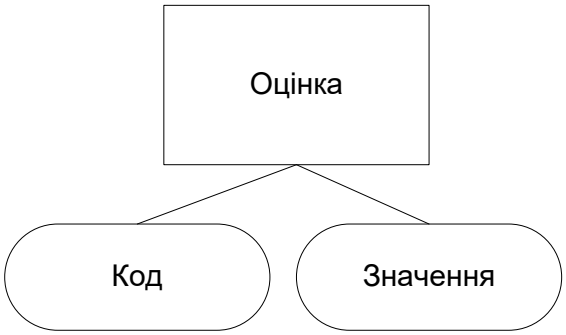


Рисунок 3.7 – Атрибути сутності «Оцінка»

Сутність «Оцінка» призначена для оцінювання працівників, після надання

ними послуг, а також вона враховуються при пошуку нових працівників для підвищення впевненості замовника, що проект розробляється надійними працівниками.

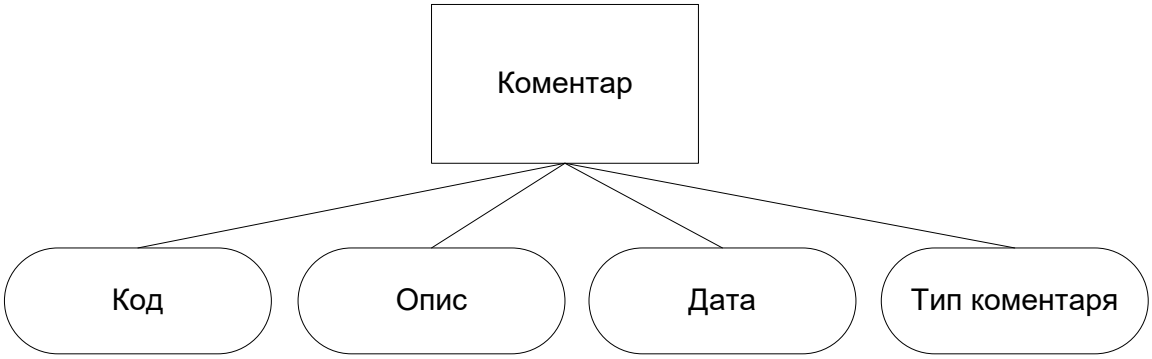


Рисунок 3.8 – Атрибути сутності «Коментар»

Зважаючи на наявність атрибутів, які можуть виступати в якості підпорядкованих сутностей, задля забезпечення цілісності даних до моделі даних додано додаткові сутності:

- тип користувача;
- тип задачі;
- пріоритет задачі;
- статус задачі;
- тип коментаря.

Атрибути додаткових сутностей зображено на рис. 3.7 – 3.9.

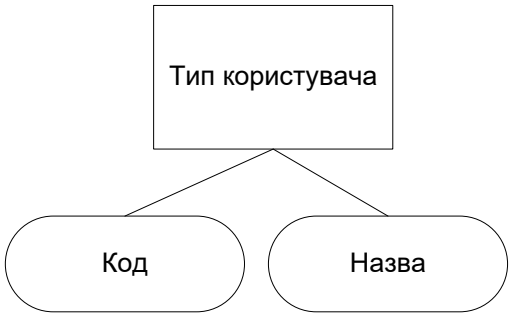


Рисунок 3.9 – Атрибути підпорядкованої сутності «Тип користувача»

Тип користувача це спеціальна сутність для розрізнення користувачів по їх

функціональним обовязкам таким як: розробник, бізнес-аналітик, замовник, DevOps інженер, тестувальник. На етапі фізичного проектування цю сутність зручно буде представити у вигляді структури перерахування.

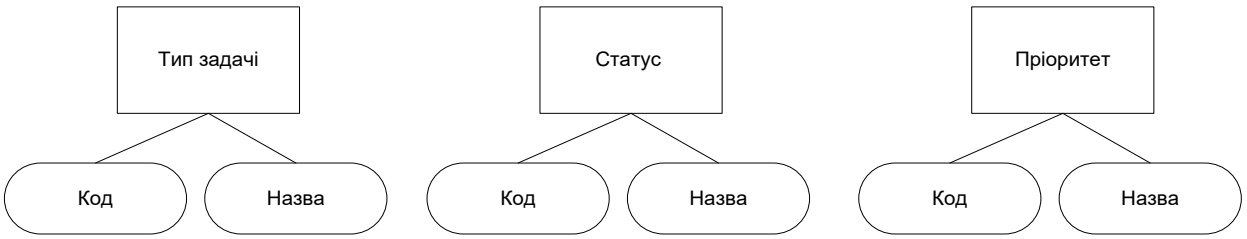


Рисунок 3.10 – Атрибути підпорядкованих сутностей «Тип задачі», «Статус», «Пріоритет»

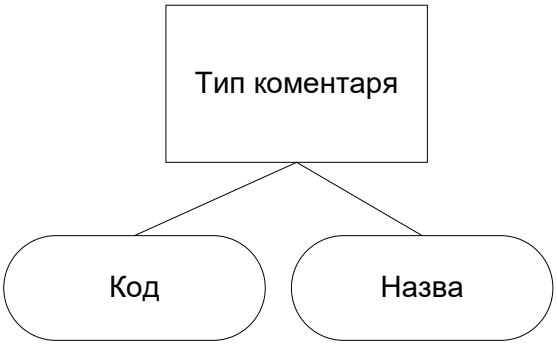


Рисунок 3.11 – Атрибути сутності «Тип коментаря»

Для подальшого визначення фізичної структури бази даних та способів адресації зовнішніх ключів згідно методу проектування «сутність-зв’язок» при побудові ER-діаграм виникає необхідність встановлення ступенів зв’язків між сутностями [13-18].

Схеми зв’язків для сутності «Користувач» наведено на рис. 3.12-3.18. Слід зазначити, що між сутностями «Користувач» та «Задача» є два види зв’язку: «Ставить» та «Виконує».

Рисунок 3.12 – ER-діаграма зв’язку сутностей «Користувач» та «Проект»

Рисунок 3.13 – ER-діаграма зв’язку «Ставить» між сутностями «Користувач» та «Задача»

Рисунок 3.14 – ER-діаграма зв’язку «Виконує» між сутностями «Користувач» та «Задача»

Рисунок 3.15 – ER-діаграма зв’язку сутностей «Користувач» та «Коментар»

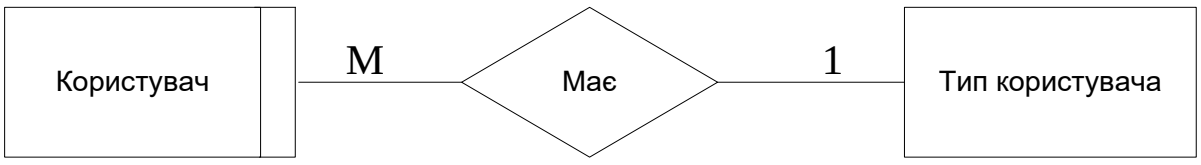


Рисунок 3.16 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

Аналогічним чином між сутностями «Користувач» та «Оцінка» є два види зв’язку: «Надає» та «Отримує» (рис. 3.17-3.18). Іншими словами користувачі

надають іншим користувачам оцінки, що необхідно для формування рейтингу працівників та замовників.

Рисунок 3.17 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

Рисунок 3.18 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

Рисунок 3.19 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

Для сутності «Вимога» спостерігається внутрішній зв'язок (рис 3.20).

Рисунок 3.20 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

На рис. 3.21-3.23 наведено зв'язок сутності «Задача» з підпорядкованими сутностями «Тип задачі», «Пріоритет» та «Статус».

На етапі фізичного проектування сутність «Задача» матиме три підпорядковані сутності у вигляді перерахувань, що дозволить зручно нею користуватися.

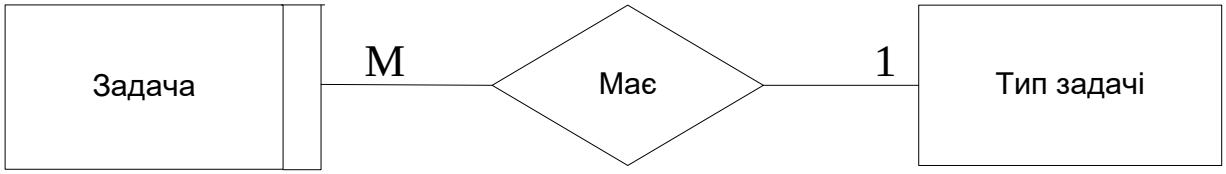


Рисунок 3.21 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

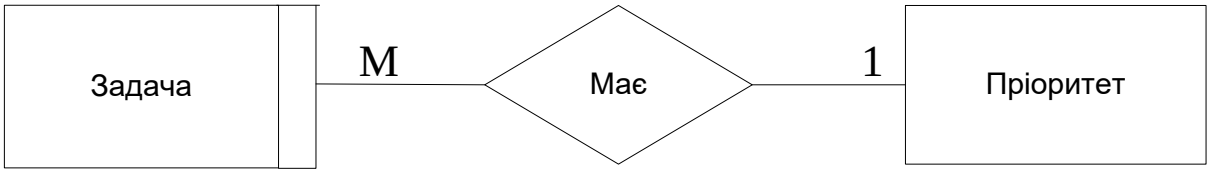


Рисунок 3.22 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

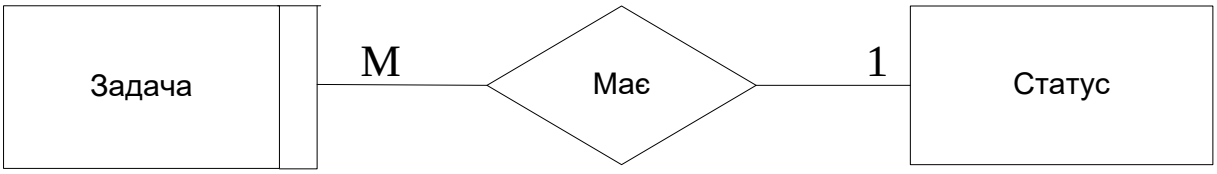


Рисунок 3.23 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

Зв'язок сутності «Коментар» з підпорядкованою сутністю яка характеризує його тип, наведено на рис. 3.24.

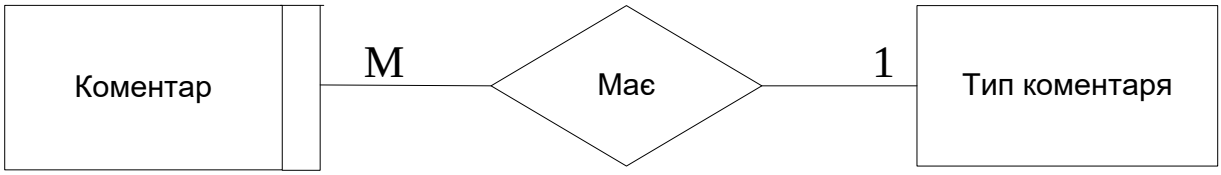


Рисунок 3.24 – ER-діаграма зв’язку сутностей «Користувач» та «Тип користувача»

В цілому ER-діаграма надає можливість побачити початкову структуру моделі даних, що будуть використовуватися в розробляємій платформі та наглядно дозволяє переглянути та проаналізувати всі сутності та їх атрибути на предмет

необхідності в даній платформі і за потребою додавання нових елементів на етапі концептуального проектування набагато легше ніж на етапі фізичного проектування бази даних, оскільки концептуальна модель даних не потребує урахування таких об’єктів як тригери та реалізацію зв’язку багато до багатьох , які є специфічними для кожної СУБД. ER діаграма бази даних, що об’єднує всі сутності та зв’язки для розробляємої платформи представлена в додатку Додаток А ER-Діаграма.

3.2.2 Фізична модель бази даних

Фізична модель бази даних розроблена на базі СУБД MS SQL Server засобами структурованої мови запитів SQL з використанням можливостей Entity Framework. Найвідомішим, функціональним і широко використовуваним ORM в світі .NET є Entity Framework [24-29].

При використанні Entity Framework в додатку існує три підходи для організації взаємодії Entity Framework з базою даних: Code-First, Model-First і Database-First.

Підхід Code-First, який вперше з'явився в Entity Framework 4.1, зазвичай використовується, коли вже є існуюча програма, що містить модель даних. Ця модель, як правило, описується за допомогою декількох класів і коду взаємодії між цими класами.

Підхід Model-First [24-29], який вперше з'явився у версії Entity Framework 4, застосовується розробниками, які не хочуть використовувати інструменти СУБД для створення та управління базами даних, а також вручну налаштовувати класи моделі EDM. Фактично це найпростіший підхід при роботі з Entity Framework. Проектування моделі відбувається в графічному дизайнері EDM середовища Visual Studio.

Робочий процес створення моделі при підході Model-First починається з проектування бази даних. Як і в разі підходу Code-First, вся робота будується навколо класу контексту бази даних. Фактично, взаємодія з базою даних в цих підходах однакова. Наприклад, для вставки об'єкта, використовується наступна

послідовність дій:

1) створити об'єкт моделі і наповнити його даними;

2) створити клас контексту, успадкований від DbContext (в підході Code-First це робиться вручну, в Model-First цей клас генерується автоматично разом з сутнісними класами);

3) додати об'єкт до бази даних, використовуючи клас контексту;

4) зберегти зміни.

Підхід Database-First, що з'явився разом с Entity Framework, дозволяє писати програми для існуючих баз даних. Бази даних в реальних додатках досить швидко стають складними і намагатися створити модель для існуючої бази даних, яку можуть зрозуміти розробники, досить важко. Ще важче написати код використання моделі, в якому відбувається взаємодія з базою даних. Багато в чому, підхід Database-First є протилежністю підходу Model-First. При підході Database-First база даних вже існує, тому розробник повинен знати, де розташована база даних, а також мати інформацію про ім'я бази даних. Проте, розробник не повинен розуміти внутрішню роботу бази даних – Entity Framework і раніше приховує внутрішню реалізацію з поля зору.

При цьому підході, робочий процес створення моделі починається зі створення і проектування бази даних. Після генерації сутнісних класів моделі з існуючої бази даних, робота з Entity Framework аналогічна підходам Code-First і Model-First [24-29]. Це означає створення об'єкта класу контексту і використання цього об'єкта для виконання необхідних завдань.

Фізична модель даних, створена засобами Entity Framework та діаграма бази даних зображені на Додатку Б.

3.3 Формування та аналіз вимог до об’єкту проектування

3.3.1 Формування та аналіз вимог

Експлуатація розробленої платформи передбачається за допомогою роботи з

базою даних на основі функціонування клієнтської програми, розрахованого на три категорії користувачів: менеджерів, розробників та тестувальників. Для реалізації функціональних можливостей платформи розроблення програмного забезпечення передбачається використання додаткових служб серверу.

Призначення і цілі створення системи – платформа розроблення програмного забезпечення позаштатними працівниками представлена базою даних для запису і зберігання всієї необхідної інформації, а також клієнтським програмним забезпеченням з інтерфейсом користувача, що дозволяє за допомогою звернення до бази даних здійснювати всі необхідні операції.

Цілі створення платформа розроблення програмного забезпечення:

– своєчасне оперативне отримання достовірної інформації щодо процесу розробки програмного продукту.

– корегування вимог до певного проекту на всіх етапах розробки програмного забезпечення.

– підвищення ефективності планування роботи з виконання вимог, висунутих до проекту.

Характеристика об'єкту проектування – об'єктом проектування є платформа розроблення програмного забезпечення позаштатними працівниками.

Платформа розроблення програмного забезпечення знаходиться в межах предметної області оперування даними про процеси, що відбуваються в процесі повного циклу розробки програмного забезпечення:

- накопичення даних про вимоги до проектів;
- можливість відстеження процесу виконання проектів;

– можливість надання оцінки розробникам та іншим учасникам процесу розробки програмного забезпечення;

– можливість відстеження функціонування проекту в умовах хмарного середовища.

Вимоги до системи в цілому – автоматизовані робочі місця платформи розроблення програмного забезпечення дозволяють виконувати всі функції,

заплановані при створенні системи [18].

Вимоги до структури та функціонування системи – функціонування означеної платформи засноване на підсистемах:

- накопичення і зберігання даних;
- обробки даних;
- аналізу даних;
- візуалізації даних.

Підсистема накопичення і зберігання даних призначена для введення і зберігання даних, що описують предметну область і використовуваних для формування документів.

Підсистеми обробки і аналізу призначені для аналізу даних, накопичених в процесі розробки програмних продуктів.

Підсистема візуалізації даних призначена для створення і формування інформаційних форм у вигляді зручному для перегляду на основі даних, отриманих шляхом звернення до бази.

Загальні відомості – найменування програмного продукту: «Платформа розроблення програмного забезпечення позаштатними працівниками».

Для функціонування системи необхідно встановити наступне програмне забезпечення:

- MS Visual Studio 2017[12-20];
- Microsoft SQL Server[12-20];

Платформа розроблення програмного забезпечення позаштатними працівниками заснована на клієнт-серверній архітектурі. Серверна частина реалізована з використанням реляційної СУБД MS SQL Server. Клієнтський веб-додаток розроблено засобами мови програмування C# в середовищі MS Visual Studio 2017.

Функціональне призначення розробки – управління процесами розробки програмного забезпечення позаштатними працівниками.

Опис логічної структури – платформа розроблення програмного забезпечення

позаштатними працівниками структурно представлена такими складовими:

- база даних;
- модуль реєстрації та входу до системи;
- модуль управління проектами;
- модуль пошуку виконавця,
- модуль управління задачами.

Клієнтська програма шляхом звернення до бази даних забезпечує можливість виконання певного набору дій, заданих користувачем.

Використовувані технічні засоби – для забезпечення коректної роботи клієнтської програми і сервера бази даних потрібно виконання наступних мінімальних вимог до апаратного забезпечення [14]:

- процесор – IntelPentium 2.0 ГГц;
- обсяг оперативної пам'яті – 512 Мб;
- наявність клавіатури і миші;
- вільний обсяг на жорсткому диску не менше 15 ГБ.

Виклик і завантаження – для початку роботи з платформою, необхідно здійснити попередню настройку сервера бази даних і клієнтського додатку.

Вхідні дані – в якості вхідних даних використовуються дані з бази даних про користувачів та створені ними проекти з розроблення програмного забезпечення різної функціональної спрямованості.

Вихідні дані – вихідні дані роботи платформи представлені у вигляді результатів виконання поставлених задач та проведення робіт з реалізації вимог до програмного забезпечення з можливістю надання коментарів користувачами та виставлення оцінок розробникам.

3.3.2 Формування вимог за допомогою діаграми прецедентів

Діаграма прецедентів передбачає наявність акторів та варіантів використання. Актор – це роль, яку виконує користувач або інша система, при взаємодії з

проектованою системою. Кожен актор має унікальне ім’я. Проектування діаграми варіантів використання розпочинається з визначення списку акторів.

Варіант використання – це кінцева одиниця взаємодії актора і системи. Сукупність усіх варіантів використання повністю визначає поведінку системи. Кожен варіант використання відноситься до деякого актора. Таке відношення означає, що цей актор ініціює цей варіант використання.

Залежно від мети виконання процедури розрізняють такі варіанти використання [7]:

– основні (базові) – забезпечують необхідну функціональність ПЗ, що розробляється;

– допоміжні – забезпечують виконання необхідних налаштувань системи і її обслуговування (наприклад, архівація інформації);

– додаткові – забезпечують додаткові зручності для користувача (як правило, реалізуються, якщо не вимагають серйозних витрат яких-небудь ресурсів ні при розробці, ні при експлуатації).

Відношення включення між двома варіантами використання в мові UML є окремим випадком загального відношення залежності, яке визначається як форма взаємозв’язку між двома елементами моделі. Зміна одного елементу моделі у відношенні залежності призводить до зміни деякого іншого елементу.

У загальному випадку залежність є спрямованим бінарним відношенням, яке зв’язує між собою тільки два елементи моделі – незалежний і залежний.

Відношення включення (include) специфікує той факт, що деякий варіант використання містить поведінку, визначену в іншому варіанті використання. Графічно це відношення позначається як відношення залежності у формі пунктирної лінії з «V»-образною стрілкою, спрямованою від залежного варіанту використання до незалежного варіанту використання. Залежний варіант використання часто називають також базовим, а незалежний – врахованим варіантом використання.

Відношення розширення (extend) в мові UML також є окремим випадком загального відношення залежності між двома варіантами використання. Відношення

розширення є спрямоване бінарне відношення, що визначає взаємозв’язок одного варіанту використання з деяким іншим варіантом використання, функціональність або поведінка якого задіються першим не завжди, а тільки при виконанні деяких додаткових умов.

Для проектування платформи для розроблення програмного забезпечення позаштатними працівниками обрано наступних акторів:

- замовник;
- розробник;
- бізнес-аналітик;
- тестувальник;
- DevOps-інженер.
- архітектор
- керівник команди.

У подальших таблицях будуть описані значимі варіанти використання що наявні у розробленій діаграмі прецедентів, яку наведено в Додатку В.

Розроблена діаграма прецедентів дозволяє наглядно побачити значиму діяльність користувачі розробляємої платформи і описує процес розроблення програмного забезпечення використовуючи ресурси позаштатних працівників, на розробленій діаграмі прецедентів (Додаток В), лише замовник є не позаштатним працівником і при необхідності замовник у випадку незадоволеності роботою позаштатних працівників може замінити того чи іншого виконавця роботи, але лише при умові порушення сроків виконання поставленого завдання, після чого невиконавший роботу працівник отримає погану оцінку рейтингу від замовника. Замовник може в будь-який момент змінити вимоги, тоді всі заплановані задачі переходять в статус очікування, а існуючі дороблюються до кінця, як було заплановано.

Але діаграма прецедентів хоч і містить значимі прецеденти, є певні прецеденти яких там не було зображено, щоб не ускладнювати і так складну для читання діаграму. Платформу для розроблення ПЗ позаштатними працівниками

потрібно підтримувати оновлювати, та стежити за підозрілою діяльністю користувачів системи, особливо це стосується нових користувачів, так користувачів з низьким рейтингом, які можуть використовувати платформу для зловмисних цілей. Цими задачами займаються спеціально навчені люди, які можуть вираховувати користувачів зловмисників, та відповідно блокувати назавжди їх доступ до платформи розроблення ПЗ позаштатними працівниками. Також цей персонал виконує обов’язки адміністрування платформи.

Оскільки платформа для розроблення ПЗ призначена для використання позаштатними працівниками

Прецедент «Створення проекту», опис якого зображено у табл. 3.1, є початковою точкою функціонування розробляємої платформи для користувача замовника.

Таблиця 3.1 – Опис прецеденту «Створення проекту»

Назва	Створити проект	
Контекст використання або мета	Створення нового проекту в платформі	
Видимість та рівень видимості для актора	Для замовника.	
Передумова	Замовник має бути зареєстрованим	
Умова успішного виконання	Всі обов’язкові поля для створення проекту введені вірно	
Умова невиконання	Будь-яке обов’язкове поле для створення проекту введене не вірно	
Актори, уразі наявності первинний, вторинний	Замовник	
Наявність тригеру	Відсутній	
Опис	Кроки	Дія
	1	Реєстрація

	2	Заповнення необхідних полів про проект
Варіанти виконання (sub variant)	-	

Описаний прецедент представляє собою створення нового проекту, що є єдиною функцією щойно зареєстрованого замовника. Надалі цей проект стає доступним замовнику і з ним в подальшому можна працювати іншим користувачам.

Прецедент «Створення вимоги», опис якого зображено у табл. 3.2, це дія що додає в проект нову вимогу.

Таблиця 3.2 – Опис прецеденту «Створення вимоги»

Назва	Створення вимоги	
Контекст використання або мета	Додавання в проект нової вимоги	
Видимість та рівень видимості для актора	Видимий для замовника, бізнес аналітика та архітектора	
Передумова	Створений проект	
Умова успішного виконання	Всі обов’язкові поля для створення вимоги введені вірно	
Умова невиконання	Будь-яке обов’язкове поле для створення вимоги введене не вірно	
Актори, уразі наявності первинний, вторинний	Первинний актор замовник, вторинні: бізнес аналітик, архітектор та керівник команди	
Наявність тригеру	Відсутній	
Опис	Кроки	Дія
	1	Вибір проекту для якого буде створена вимога
	2	Заповнення обовязкових полів для

		створення вимоги (опис, пріоритет)
Розширення (extention)	Призначити виконавця вимоги, створити підвимогу.	
Варіанти виконання (sub variant)	Затвердити вимогу, переглянути вимогу, редагувати або видалити вимогу, змінити ціну	

Розроблення ПЗ використовуючи робочий процес розробляємої платформи починається з створення та обробки вимог, вимогу створює замовник, після цього її переглядає і редагує бізнес-аналітик, узгоджує ціну з замовником, і після цього вимогу обробляє архітектор, а далі вимогу обробляє керівник команди.

Прецедент «Створення задачі», опис якого зображено у табл. 3.3, це дія що додає до вимоги нову задачу.

Таблиця 3.3 – Опис прецеденту «Створення задачі»

Назва	Створення вимоги	
Контекст використання або мета	Додавання в проект нової задачі	
Видимість та рівень видимості для актора	Видимий для керівника команди, бізнес-аналітика	
Передумова	Створена вимога для якої буде створюватися задача	
Умова успішного виконання	Всі обов’язкові поля для створення задачі введені вірно	
Умова невиконання	Будь-яке обов’язкове поле для створення задачі введене не вірно	
Актори, уразі наявності первинний, вторинний	Керівник команди, розробник, тестувальник	
Наявність тригеру	Кроки	Дія

	1	Вибір вимоги для якої буде створена задача
	2	Заповнення обовязкових полів для створення задачі (опис, пріоритет)
Розширення (extention)	Призначити виконавця задачі, створити підзадачу.	

Прецедент «Створення помилки», опис якого зображено у табл. 3.4, це дія що додає в до задачі нову помилку.

Таблиця 3.4 – Опис прецеденту «Створення помилки»

Назва	Створення помилки	
Контекст використання або мета	Додавання до задачі нової помилки	
Видимість та рівень видимості для актора	Видимий для керівника команди, тестувальника, розробника	
Передумова	Створена задача для якої буде створена помилка	
Умова успішного виконання	Всі обов’язкові поля для створення помилки введені вірно	
Умова невиконання	Будь-яке обов’язкове поле для створення помилки введене не вірно	
Актори, уразі наявності первинний, вторинний	Тестувальник, розробник, керівник команди	
Наявність тригеру	Відсутній	
Опис	Кроки	Дія
	1	Вибір задачі
	2	Заповнення обовязкових полів для створення помилки (опис, пріоритет)
Розширення (extention)	Призначити виконавця помилки	

Варіанти виконання (sub variant)	Редагувати помилку, видалити помилку, переглянути помилку, верифікувати помилку

Під час тестування задачі відбувається додавання помилок в систему управління проектом платформи для розроблення ПЗ позаштатними працівниками, далі розробники редагують статус помилки і переводять в статус вирішено, а після цього тестувальники верифікують помилку і переводять в статус зроблено, в разі успішного усунення помилки.

3.3.3 Формування та аналіз вимог за допомогою діаграми комунікації

Використання діаграми комунікації дозволяє наглядно побачити можливості комунікації користувачів між собою та значимими модулями платформи, а також комунікацію модулів між собою. Так на діаграмі комунікації наведеної на рисунку 3.24, можна побачити комунікації значимих модулів та користувачів системи, і переконатися що користувачі комунікують лише з необхідними їм користувачами і модулями, а не з усіма підряд, що було б порушенням сучасних парадигм проектування програмних систем, а саме першої парадигми об’єктно орієнтованого програмування – інкапсуляції. На діаграмі комунікації (рисунок 3.24) можна побачити, що комунікація з сервером бази даних не відбувається напряму з будь-яким з користувачів, а відбувається через певний модуль, відповідно така є особливість передбачена для користувачів системи, щоб вони комунікували через різні модулі, для можливого логування їх повідомлень та дій, для можливості розібратися в помилках, якщо вони виникнуть, та для полегшення тестування системи при розробленні системи та її тестуванні, також подібна структура дозволяє уникнути проблем повязаних з виходом одного з модулів з ладу, якщо з ладу вийде лише один з модулів – інші працюватимуть коректно. На діаграмі комунікації можна побачити що клієнт комунікує лише з вимогою та проектом, використовуючи веб-додаток, це зроблено для того, щоб клієнт він же замовник, працював в

основному з вимогами та не бачив що саме відбувається всередині його найманої команди. Такий підхід дозволяє замовнику не поглиблюватися в процеси розроблення ПЗ, а виконавцям дозволяє бути більш спокійними і не переживати, що за ними постійно наглядають, що в свою чергу дозволяє пришвидшити процес розроблення ПЗ, інкапсулюючи різні категорії користувачів один від одного. Подібний підхід використовується і для інших учасників платформи для розроблення ПЗ позаштатними працівниками.

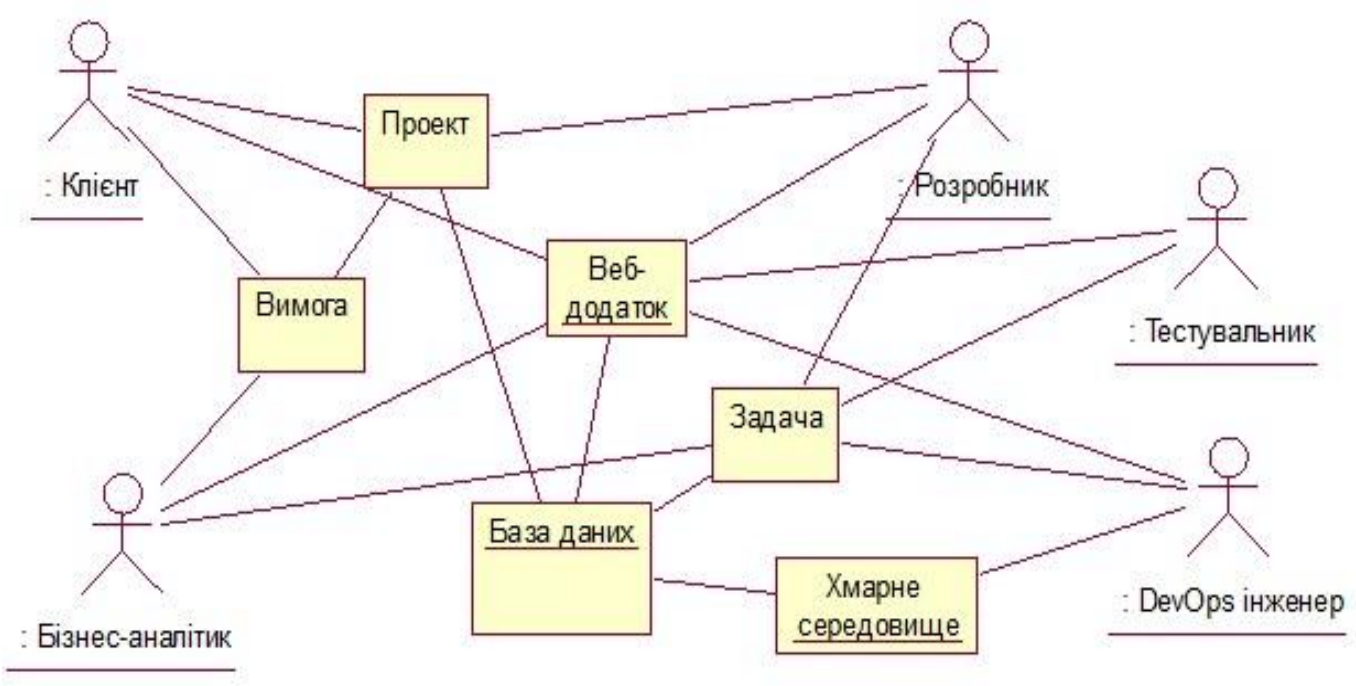


Рисунок 3.24 – Діаграма комунікації

3.3.4 Аналіз вимог за допомогою діаграми послідовності

Діаграма послідовності є одним з головних елементів проектної документації на програмне забезпечення та найбільш повно описує послідовність протікання процесів притаманних розроблюваній системі. Фокус управління призначений для виділення активності об’єктів і зображується на діаграмі у вигляді вузького прямокутника, верхня сторона якого є початком отримання фокусу управління об’єкту (початок активності), нижня сторона – закінченням фокусу управління

(закінчення активності). Періоди активності можуть чергуватися з періодами пасивності або очікування (декілька фокусів управління) [14].

Ініціатором взаємодії в системі може бути актор або зовнішній користувач. Актор може мати власне ім’я або залишатися анонімним. Найчастіше актор і його фокус управління існують в системі постійно.

Діаграму послідовності наведено в Додатку Д.

Приведена діаграма послідовності надає можливість чітко побачити робочий процес платформи для розроблення програмного забезпечення позаштатними працівниками. Замовник створює проект і разом з бізнес аналітиком формує вимоги, бізнес аналітик перевіряє коректність вимоги вносить правки, узгоджує з замовником і формує задачі для розробників, розробники розробляють ПЗ, проводять код ревю, у разі виявлення помилок проходять повторне код ревю і віддають задачу тестувальнику, тестувальник тестує і в разі виявлення помилок, заводить в системі управління проектами нову помилку, яку розробники лагодять, після цього тестувальник повертається до ранішествореної помилки, і в разі успішного її вирішення верифікує її. Після верифікації всіх помилок, та вирішення всіх завдань, вимога з якою пов’язані вирішені задачі автоматично вирішується і передається DevOps інженеру для розгортання проекту в хмарному середовищі, замовнику приходить сповіщення про вирішення тої чи іншої вимоги і він може користуватися розробленим рішенням, після чого коли замовнику знадобиться новий функціонал він знову звертається до бізнес аналітика і цикл розроблення продовжується.

4 РЕАЛІЗАЦІЯ ПЛАТФОРМИ ДЛЯ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОЗАШТАТНИМИ ПРАЦІВНИКАМИ

4.1 Архітектурне проектування програмного забезпечення

При розробленні програмного забезпечення використано шаблон «модель-подання-контролер» (Рисунок 4.1), який лежить в основі програмної технології ASP.NET MVC обраної для реалізації платформи.

Рисунок 4.1 – Шаблон «модель-подання-контролер»

Модель може являти собою об'єкт, який реалізує перемикач. У найпростішому випадку дана модель характеризується станом: вимкнений або включений, і, крім того, дозволяє змінювати його – об'єкт моделі має метод зміни стану: вимкнути і включити. Умовно кажучи подання відображає на дисплеї стан перемикача за допомогою певної текстової або графічної форми. Наприклад, візуальний елемент може відображати текстову мітку, яка при зміні стану моделі перемикача відобразить відповідний статус задачі: «виконано» чи «не виконано». Крім відображення поточного стану такий спосіб організації архітектури програмного забезпечення дозволяє користувачеві змінювати стан перемикача за допомогою графічних примітивів, наприклад, двох з написами: «Виконати задачу» і «Задача

виконана». Умовне подання вміє тільки відображати стан моделі перемикача, для зміни подання звертається до контролера. Контролер в свою чергу являє собою об'єкт, який вміє змінювати стан моделі перемикача.

Для роботи з базою даних використовується шаблон unit of work, який реалізується через ORM Entity Framework. Unit of work це шаблон призначений для створення єдиного методу роботи з сховищем даних, в якого є загальні методи для роботи з абстрактним сховищем даних, що дозволяє у разі потреби досить просто перейти до використання іншого провайдера баз даних відмінного від MS SQL Server, або навіть використати не реляційну базу даних, при цьому класи бізнес логіки змінювати буде непотрібно, а лише реалізувати всі методи інтерфейсу Unit of work.

Розроблення платформи відбувалось з використанням обєктно орієнтованого дизайну SOLID.

Перший принцип SOLID а саме принцип єдиного обовязку, був реалізований в платформі в більшості обєктів, наприклад в тому ж самому unit of work, оскільки це обєкт який займається виключно роботою з сховищами даних.

Діаграма компонентів для розроблюваної системи наведена в додатку Додаток Е Діаграма компонентів. Розробляєма платформа складається з кількох підсистем, що так чи інакше посилаються одна на одну, це можна побачити на діаграмі компонентів яка складається з таких компонентів:

- база даних;
- ORM Entity Framework;
- модуль реєстрації;
- модуль пошуку працівників;
- клієнтський додаток;
- модуль управління проектами;
- модуль аукціону вимог;
- модуль роботи з хмарними сервісами;
- модуль обміну повідомленнями;

– модуль реєстрації.

Фізичне розміщення платформи що розробляється можна подивитися на додатку Ж.

На діаграмі розгортання розміщаються тільки ті елементи фізичного представлення моделі, які існують під час виконання програмної системи. Ті елементи, які не використовуються на етапі виконання, на діаграмі розгортання, як правило, не показуються. Так, наприклад, компоненти з текстами програм можуть бути присутніми на діаграмі компонентів, а на діаграмі розгортання вони не вказуються.

Слід зазначити, що на відміну від діаграм логічного представлення і діаграм компонентів, діаграма розгортання проектується, як правило, в єдиному екземплярі, оскільки вона повинна цілком представляти особливості топології і реалізації усієї системи [14].

Фізично платформа що розробляється представляє собою п’ять сервісів що можуть розміщуватися на різних комп’ютерах, а саме:

– сервер з сервісом;
– клієнтський додаток;
– сервіс сповіщень;
– шина повідомлень;
– сервер бази даних.

Кожен з сервісів може дублюватися для забезпечення стабільності або обробки більшої кількості запитів від користувачів того чи іншого сервісу.

Сервісний сервер представлений на рисунку 4.2 представляє собою основний модуль в якому знаходиться бізнес логіка платформи, він містить абстракцію для робити з базою даних, робота з базою відбувається лише через цей сервіс. Сервер з сервісом для взаємодії з іншими сервісами містить MVC контролери та RPC API. RPC API використовується для роботи з компютерами в локальній мережі по TCP протоколу. RPC API створений для подальшого розвитку платформи розроблення програмного забезпечення, коли вона буде використовуватися для штатних

працівників. Робота в локальній мережі потрібна буде, оскільки великі компанії нікому окрім себе не довіряють і встановлюють ПЗ лише локально на власних серверах. MVC контролери використовувати набагато зручніше і вони є основою побудови платформи що розробляється.

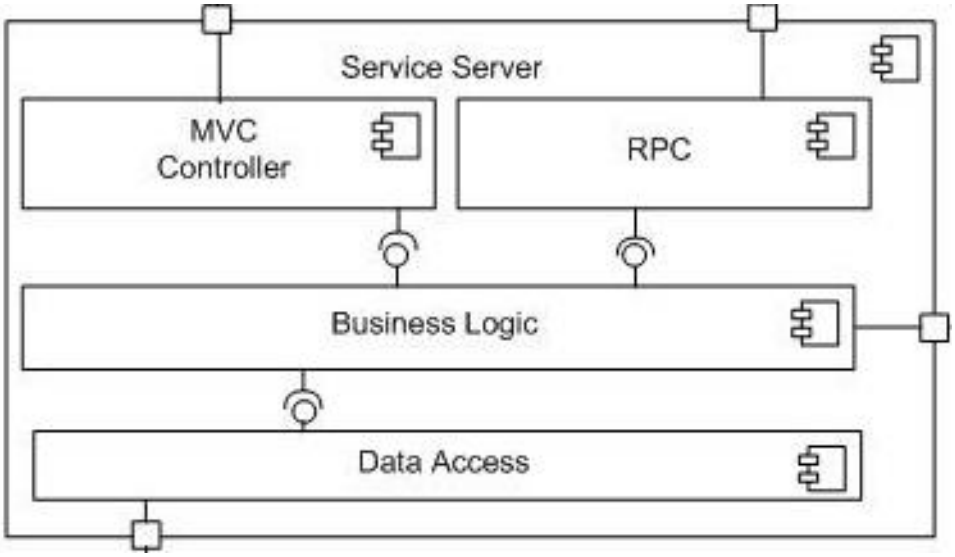


Рисунок 4.2 – Сервісний сервер

На рисунку 4.3 зображено представлення клієнтського додатку, він спроектований таким чином, що це може бути не лише браузер який комунікує з MVC контроллером, це може бути будь який додаток що реалізує RPC інтерфейс, як вже було сказано це було зроблено для подальшого застосування платформи для розроблення програмного забезпечення у великих компаніях.

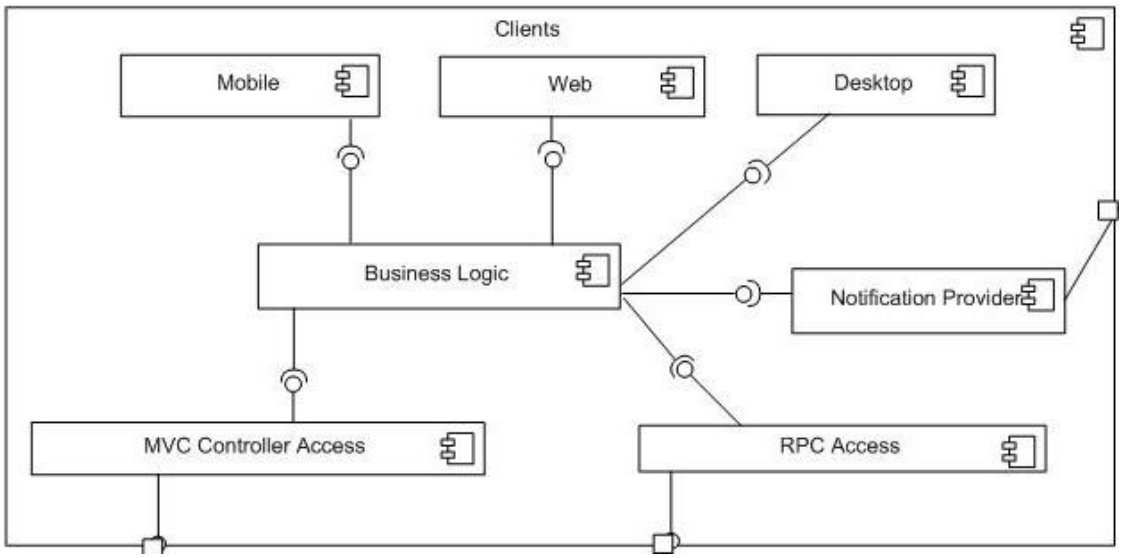


Рисунок 4.3 – Клієнтський додаток

На рисунку 4.4 зображено сервіс сповіщень, який призначений для оповіщення клієнтів про певні події, наприклад про створення нового проекту який може зацікавити розробника. Сервісний сервер надсилає до шини повідомлень повідомлення про створення нового проекту, шина повідомлень надсилає повідомлення в сервіс сповіщень і сервіс сповіщень у певний час, або коли набереться якась кількість нових цікавих розробнику проектів, надсилає йому електронний лист зі списком проектів. Без сервісу сповіщень можна було обійтися, але в такому випадку розробник отримував би новий електронний лист при кожному створенні нового проекту якимось із замовників, і відповідно розробник не хоті в би отримувати зайві повідомлення на свою електронну скриньку.

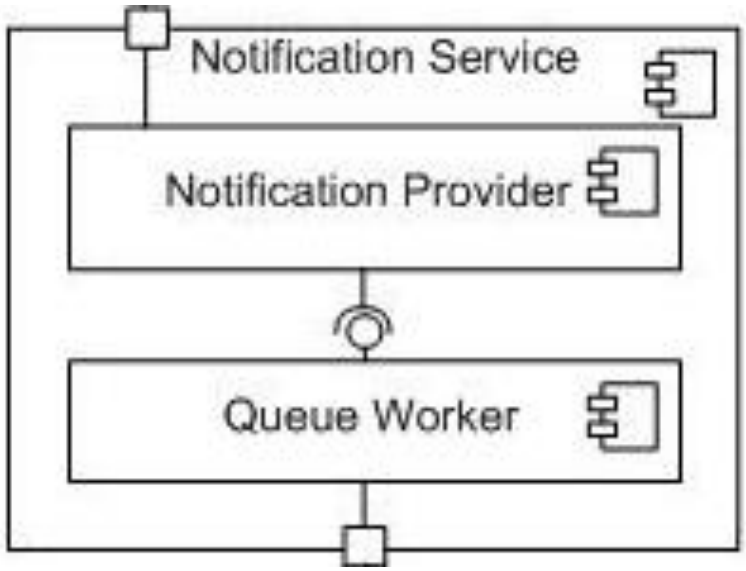


Рисунок 4.4 – Сервіс сповіщень

На рисунку 4.5 представлена шина повідомлень, яка призначена для обміну повідомленнями між різними компонентами платформи. ЇЇ завдання це отримання повідомлень і їх гарантована доставка, якщо повідомлення не було відправлене воно через певний час буде надіслане повторно, а через задану кількість невдалих спроб, потрапить в спеціальну чергу невідправлених повідомлень, звідки можна буде надіслати їх вручну. Завдяки використанню шини повідомлень платформа що розробляється реалізує подійно орієнтовану парадигму програмування, яка надає один з найкращих способів оброблення помилок, оскільки якщо відбулась помилка при відправці повідомлення, отримувач повідомлення не зламається і зможе отримувати будь-які інші повідомлення, і навпаки якщо зламається один з отримувачів повідомлення, всі інші отримувачі і відправник працюватимуть.

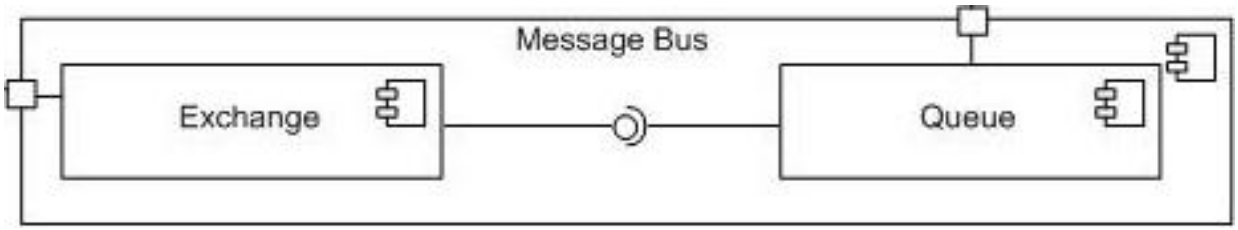


Рисунок 4.5 – Шина повідомлень

На рисунку 4.6 представлено сервер баз даних, що містить відповідно дані та

певний сервіс, що відповідає за роботу з даними і з допомогою якого відповідно проводяться дії над даними. Сервіс що працює з даними проводить кешування результатів, індексацію вставлених даних для подальшого швидкого доступу до них, відповідає за балансування навантаження та резервне копіювання, у випадку використання декількох екземплярів.

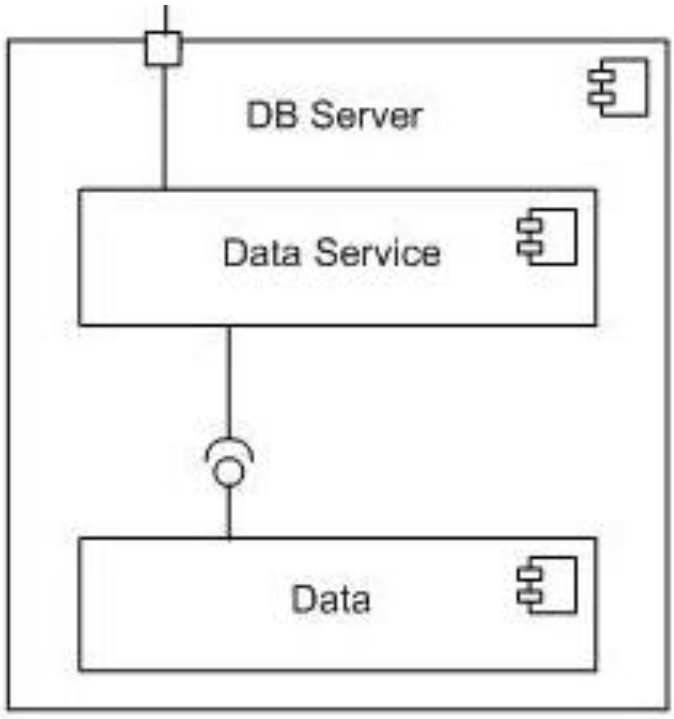


Рисунок 4.6 – Сервер бази даних

Загальну структуру платформи можна переглянути в додатку Г. На структурній схемі, з додатку Г зображено чотири основних компонентів платформи що розробляється, які можуть дублюватися, для підвищення продуктивності або безпеки збереження даних. Також невідємним компонентом є користувач, який працює з платформою через веб браузер, на сучасних мобільних та десктопних операційних системах. Основні компоненти мають такзвані балансувальні підсистеми, позначені на дівграмі як «балансувальники навантаження», вони призначені для того щоб розподіляти навантаження між різними екземплярами компоненту з якими вони повязані (Рисунок 4.7).

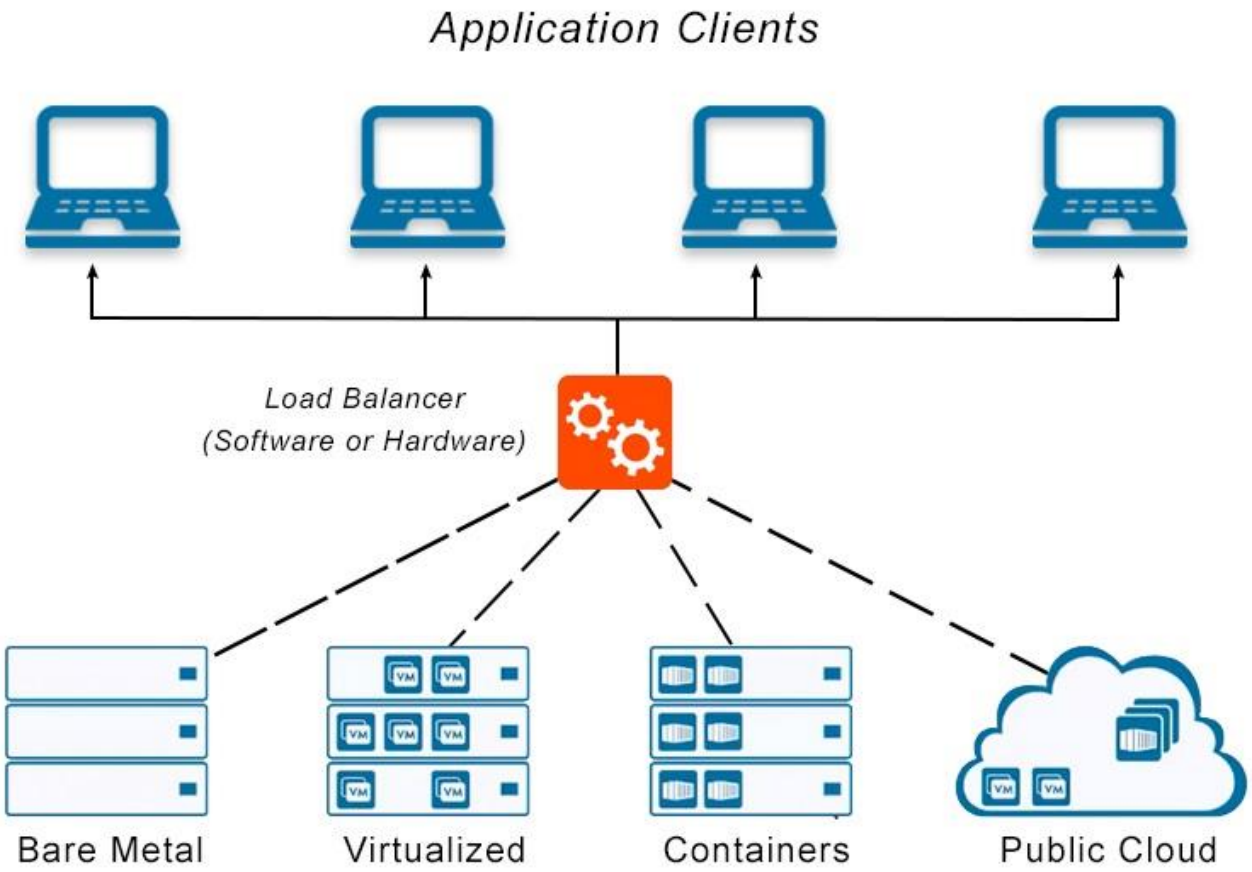


Рисунок 4.7 – «Балансувальник навантаження»

На рисунку представлено абстрактний балансувальник навантаження, який може працювати як з хмарним середовищем, так і з фізичним. Використовуючи хмарне середовище, можна в залежності від навантаження додавати нові екземпляри програми у разі необхідності і в такому випадку яке б навантаження не було, користувачі завжди зможуть користуватися системою на відміну від підходу з локальним розміщенням серверів, коли їх певна кількість. В цьому випадку апаратне забезпечення або простоюватиме у разі невеликої кількості користувачів, або буде перевантажене у разі завеликої кількості користувачів.

4.2 Розроблення користувацького інтерфейсу

Графічний інтерфейс веб-додатку платформи розроблення програмного забезпечення позаштатними працівниками наведено на рис. 4.8. Вікно входу має

стандартний набір функцій та передбачає можливість реєстрації в системі. Логін вводиться в звичайне текстове поле, а поле паролю містить функціонал приховування введених даних, маскуючи їх символом крапки.

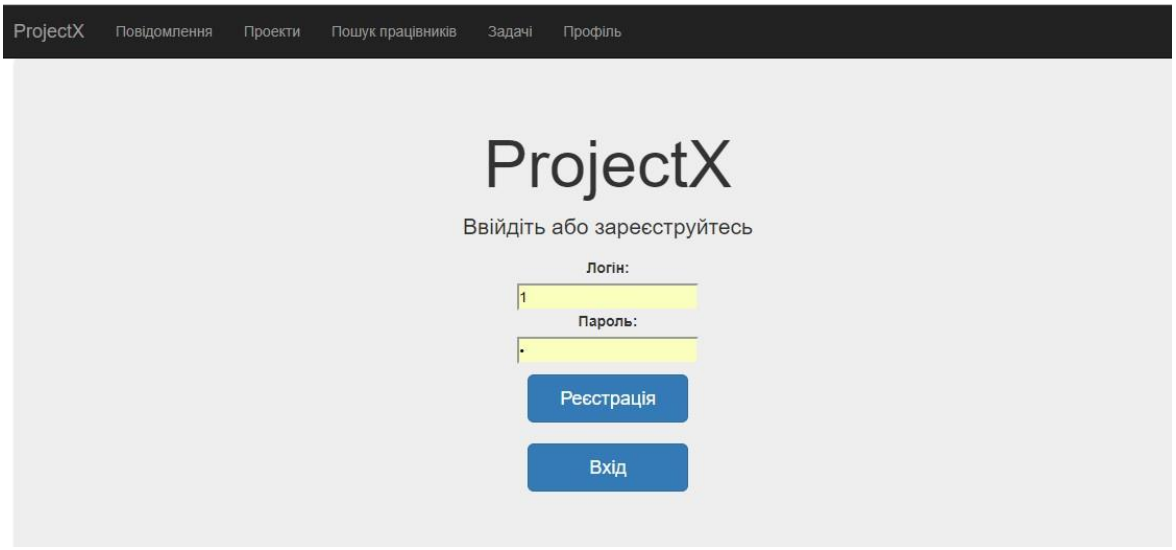


Рисунок 4.8 – Вікно входу

Головне меню додатка складається з пунктів «Повідомлення», «Проекти», «Пошук працівників», «Задачі» та «Профіль». Воно доступне в горі екрану, оскільки зазвичай людям складно тримати в голові багато значень, меню містить лише п’ять пунктів, які легко запам’ятовуються типовому користувачу.

Користувач моє можливість надсилати повідомлення іншим користувачам платформи, за допомогою пункту «Повідомлення» представленого на рисунку 4.9.

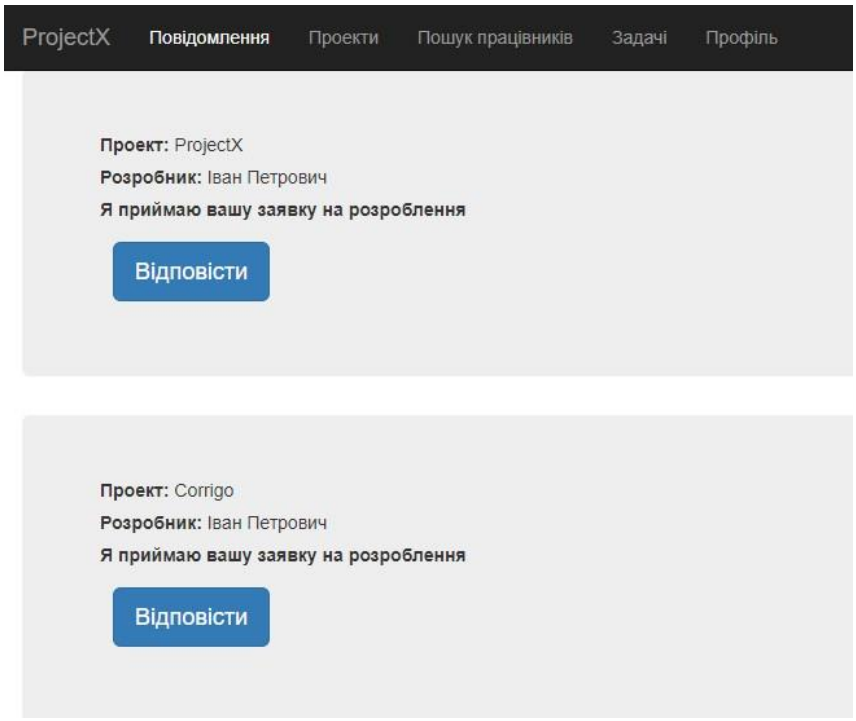


Рисунок 4.9 – Вікно формування повідомлень

При створенні повідомлення є можливість вказати до якого проекту воно належить, а також обрати тип завдання (рис. 4.10).

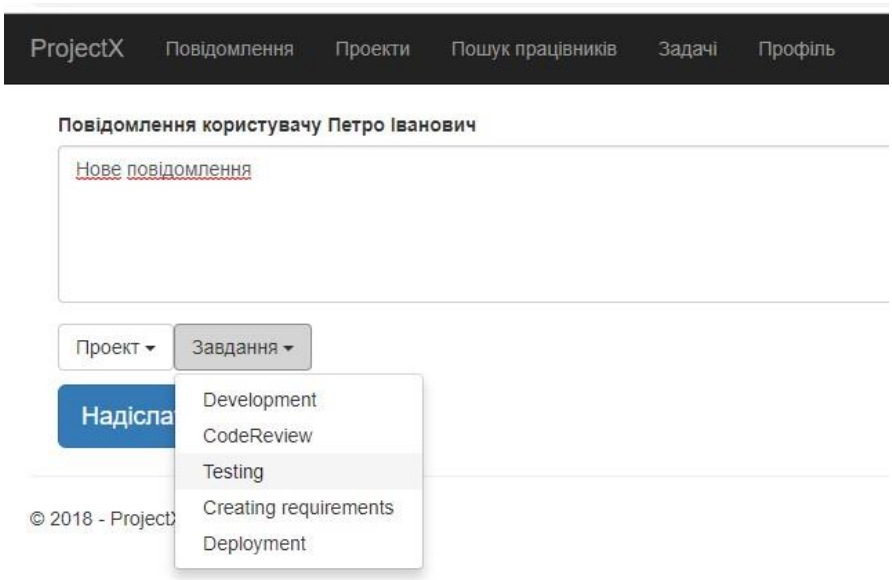


Рисунок 4.10 – Вікно введення повідомлень

Пункт меню проекти дозволяє переглянути повязані з користувачем проекти та створити новий рисунок 4.11

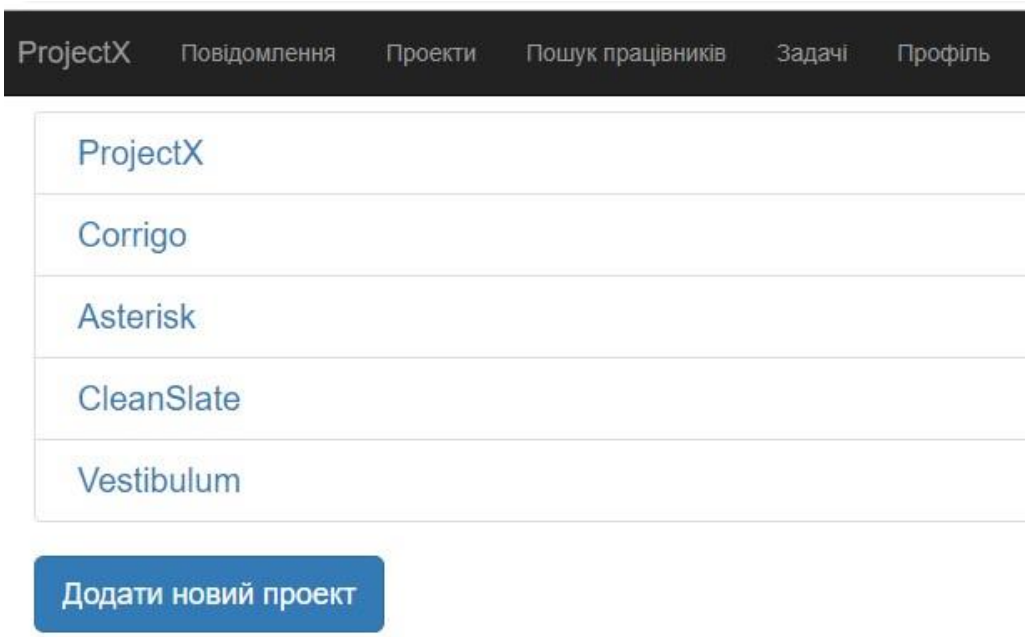


Рисунок 4.11 – Вікно додання проекту

При доданні проекту передбачена можливість повного опису проекту (рис. 4.12).

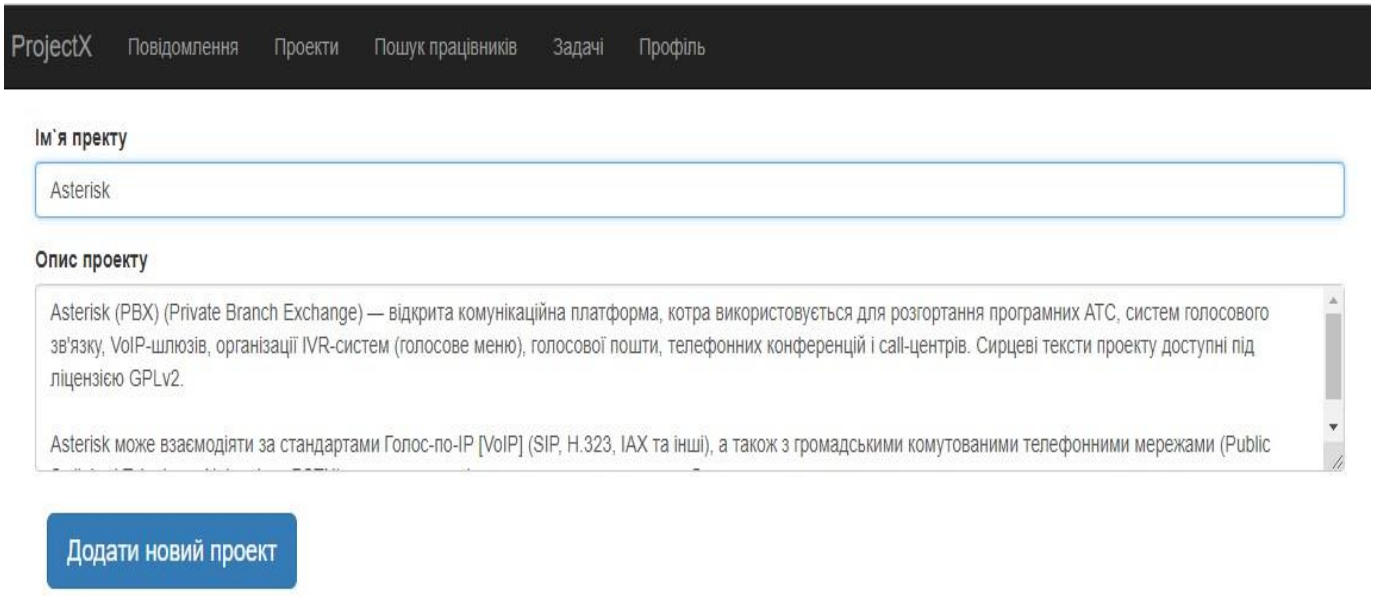


Рисунок 4.12 – Вікно опису проекту

Меню «Пошук працівників» передбачає виведення інформації про працівника з можливістю почати листування рис. 4.13. Для управління своїми задачами є вікно роботи з задачами рис.4.14. При управлінні задачами існує можливість завдання

пріоритету та статусу кожної задачі, як показано на рис. 4.15.

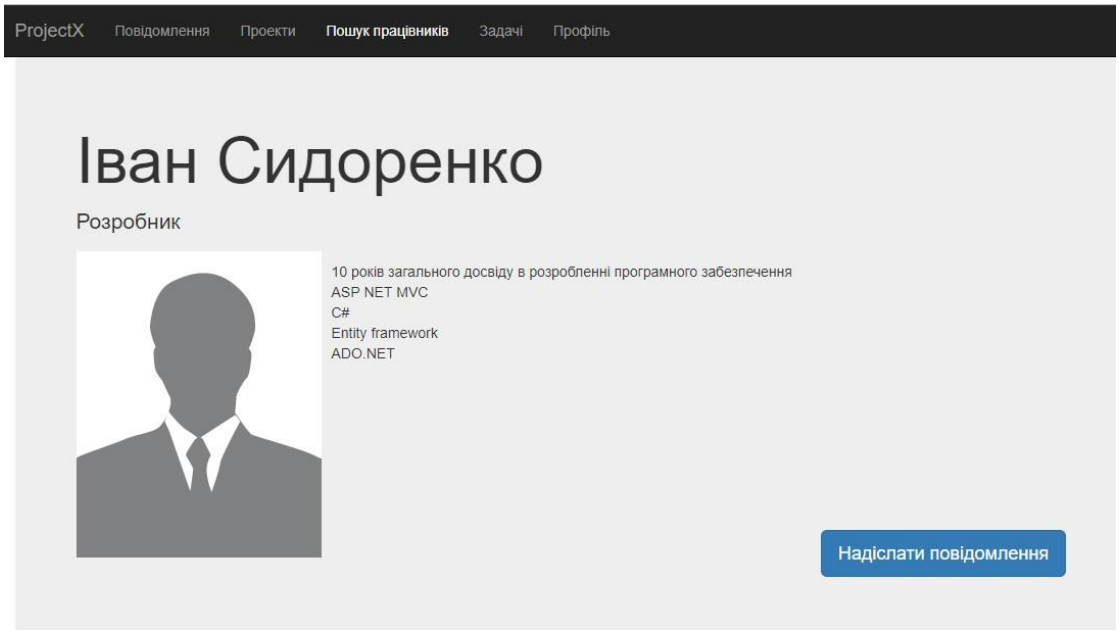


Рисунок 4.13 – Вікно пошуку працівників

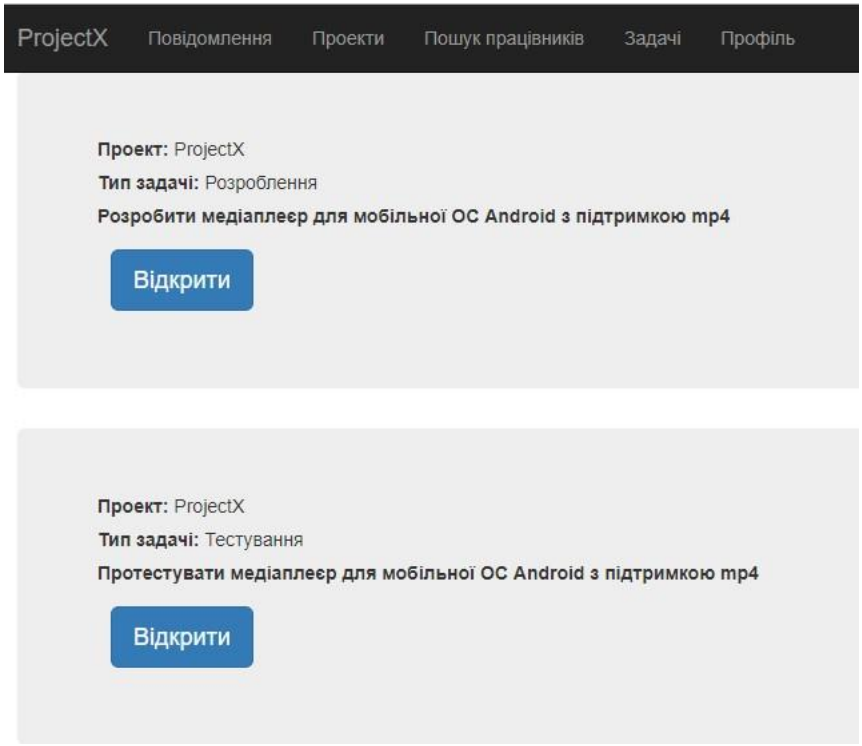


Рисунок 4.14 – Вікно роботи з задачами

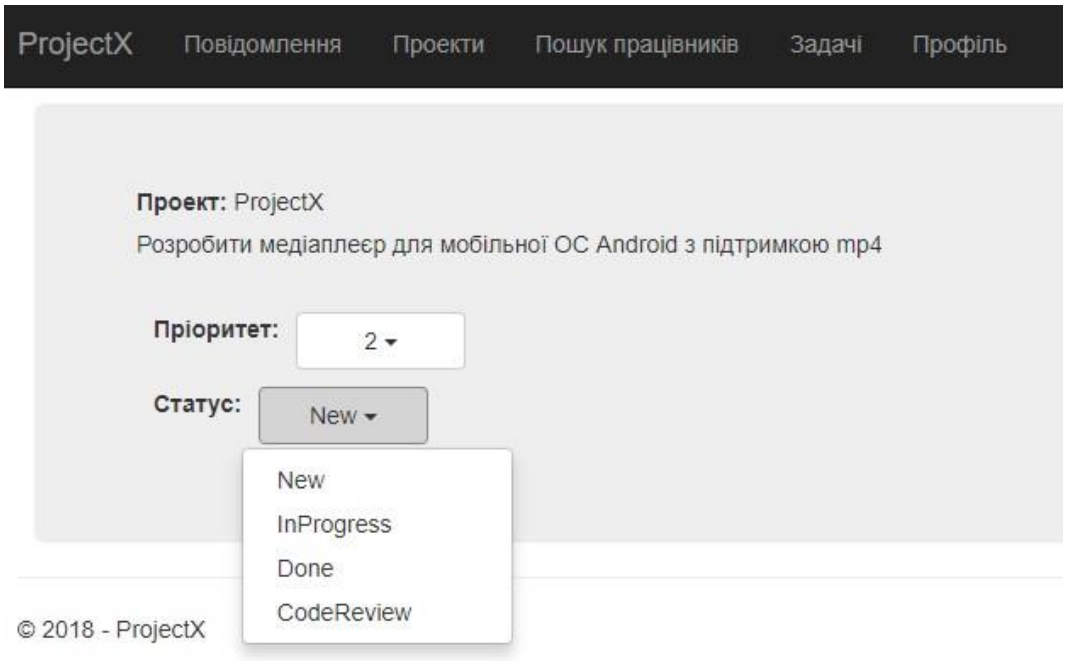


Рисунок 4.15 – Вікно завдання статусу та пріоритету задачі

Профіль користувача має стандартний вигляд з можливістю завантаження фото (рис. 4.16).

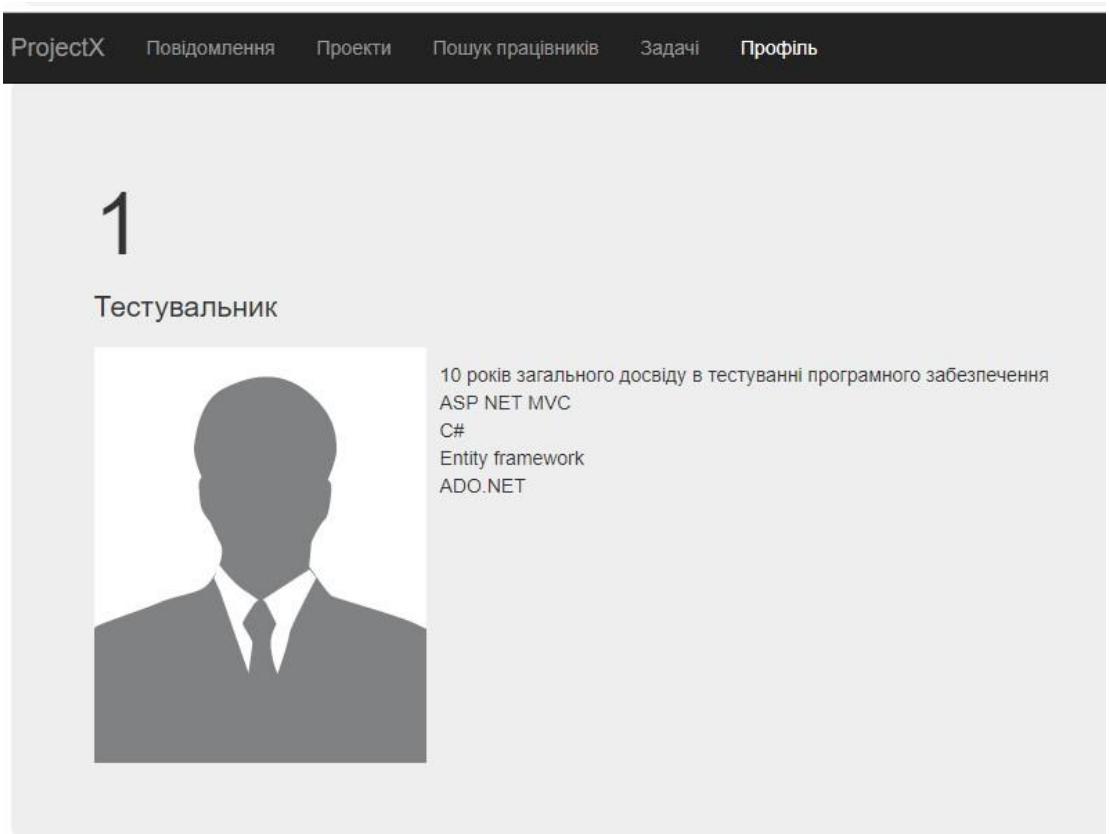


Рисунок 4.16 – Профіль користувача

Зважаючи на зазначене можна зробити висновок про достатню функціональність та ергономічність розробленого користувацького інтерфейсу, який передбачає виконання учасниками процесу створення програмного забезпечення своїх функцій в повному обсязі, про що свідчить використання адаптивного дизайну, що дозволяє змінювати інтерфейс залежно від розміру екрану, тобто інтерфейс виглядає однаково чудово як на екрані мобільного телефону, так і на моніторі комп'ютера. Графічний інтерфейс містить невелику кількість об'єктів на екрані, що дозволяє користувачам ефективно їх використовувати і пам'ятати про їх призначення.

Розроблена платформа містить значну частину бізнес логіки, яка була розподілена по різним сутностям відповідно до парадигм обєктно орієнтованого програмування, переглянути розроблені сутності можна на діаграма класів, що наведена в додатку З.

5 СТАРТАП ПРОЕКТ

У сучасних умовах все більше підприємств розглядають застосування інформаційних технологій як можливість зростання ефективності бізнесу. Сучасний ринок створює запит на виникнення і розвиток стартапів. Стартап або стартап-компанія (від англ. Start-up – запускати) являє собою нещодавно запущений проект, мета якого – в найшвидші строки окупити вкладені в нього інвестиції і отримати прибуток. Питання грошової оцінки є одним з основних при запуску нового проекту. При неправильному розрахунку бюджету проекту можна не тільки втратити частину доходу, але і навіть втратити кошти. Важливо надати не тільки очікувану оцінку стартапу але і його поточну вартість. Щоб уникнути збиткових ситуацій проводять попередню оцінку проекту і на її основі приймається рішення про реалізацію проекту або про його відхилення.

Платформа розроблення програмного забезпечення є сукупністю програмних засобів, використання яких призведе до економії часу та ресурсів з розробки програмного забезпечення.

Об'єктивними причинами використання платформи розроблення програмного забезпечення в якості стартапу є:

1) Потреба в економії часу на пошук розробників, формування та змінення вимог, тестування програмного продукту.

2) Прискорення та спрощення пошуку необхідної для прийняття організаційних рішень інформації в будь-який момент.

3) Планування та прогнозування потреби в людських ресурсах в процесі розробки програмного забезпечення.

Можна виділити такі функції стартап-проекту платформа розроблення програмного забезпечення:

- облік результатів діяльності користувачів;
- планування потреб у ресурсах;

– управління надбанням і обліком ресурсів;

– оцінка, навчання і розвиток учасників процесу розробки;

– управління мотивацією учасників процесу розробки (нормування робочого часу працівників з відповідним фінансовим і нефінансовим стимулюванням).

Окремо доречно виділити функцію інформаційного забезпечення, завдяки якій в системі накопичується інформація про діяльність системи в рамках стартап-проекту, а також на основі накопиченої інформації формується база знань.

Якісна розробка стартап-проекту є неможливою без достатнього, своєчасного і повного інформаційного забезпечення. Функцію зв'язку і засоби підтримки процесів в стартап-проекті виконує інформація.

Розробка програмного забезпечення пов'язана з постійним перетворенням інформації на кожному етапі цього процесу. На процес розробки програмного забезпечення впливають фактори, які зумовлюють його ефективність. До таких факторів належать зміни в зовнішньому середовищі, взаємопов'язаність процесів та рішень тощо. Інформаційні обмеження полягають перш за все в тому, що не всі дані, які надходять із зовнішнього середовища і циркулюють всередині проекту, можна вважати корисними. Організація системи інформаційного забезпечення має постійно, стало і безперервно підтримувати можливість надання різної інформації.

В цілому, функціонування платформи розробки програмного забезпечення має відповідати умовам оптимальності за змістом, формою і множиною факторів, що забезпечують інформаційні потреби конкретних користувачів при постійних змінах у внутрішньому та зовнішньому середовищі.

Стартап-проект платформи розроблення програмного забезпечення є найбільш ефективним інструментом створення якісного інформаційного забезпечення та налагодження інформаційних комунікацій в процесі роботи над проектами [1, 3]. Затребуваність такого програмного продукту, перш за все, залежить від його досконалості, яка надає можливість автоматизації виконання функцій при роботі з програмним забезпеченням. Досить багато стартап-проектів містять дуже хороші ідеї які з часом перетворюють наше життя кардинальним чином, причому майже всі

сучасні стартап-проекти не можуть обійтися без галузі інформаційних технологій.

Таблиця 5.1 – Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Обєднання всього процесу розроблення програмного забезпечення в один єдинй продукт	Розробка програмного забезпечення	Економія часу та ресурсів
	Супроводження програмного забезпечення	Економія часу та ресурсів
	Тестування програмного забезпечення	Економія часу та ресурсів
	Менеджмент програмного забезпечення	Економія часу та ресурсів

Основні вимоги до стартап-проектів, як правило, полягають у наступному:

– перевірити наявність аналогічних систем;

– надати інформацію про стартап-проект споживачам та партнерам в необхідному обсязі;

– надати інформацію про стартап-проект в потрібний час;

– забезпечити процеси обробки інформації при реалізації проекту;

– зберегти результати обробки інформації в необхідному вигляді.

Загальні обмеження на роботу з інформацією можуть бути представлені як група характеристик, які дійсно належать сторонам стратап-проекту, а саме:

– суб'єкту (активній стороні діяльності, одержувачу і перетворювачу інформації);

– медіатору (засобу отримання, обробки, виробництва і передачі інформації, «посереднику» між суб'єктом і об'єктом);

– об'єкту (джерелу інформації).

Складність отримання інформації на сьогоднішній день обумовлена її розподіленням та фрагментарністю, а також різнопрофільним наповненням.

Розподілення інформації означає, що доступні відомості знаходяться, умовно

кажучи, в «різних місцях». Показник різнопрофільності інформації полягає в значній кількості використовуваних джерел, оброблюваних сфер знання, методів [5, 9].

Існування інформації в розподіленому і різнопрофільному видах представляє значну складність для її отримання та обробки. Відповідно, будь-яка робота зі збору, передачі, узагальнення, аналізу, зберіганню інформації завжди буде виходити з неповноти (відкритості) інформації і, отже, буде суб'єктивно забарвленою. Рівень, повнота і характер наданої інформації визначають повноту і характер рішення з певного питання. Таким чином, найбільш значущими в цих умовах факторами роботи з інформацією є:

– фрагментарність будь-якої первинної інформації;
– різнохарактерність будь-якої первинної інформації;
– суб'єктивність будь-якої узагальненої інформації.

Стартап-проекти лише тоді виправдовують своє призначення і фінансові витрати на їх функціонування, якщо вони:

– забезпечують безперервну роботу з інформаційними ресурсами (їх збір, обробку, зберігання, копіювання, верифікацію і актуалізацію);
– забезпечують постійний аналіз інформації з предметів свого ведення, контролюють її наявність і своєчасне оновлення в базах даних, виявляють тенденції, що складаються в сфері життєдіяльності стартап-проекту;
– підтримують виконання операцій аналітичної обробки масивів даних в рамках вирішуваних завдань;
– сприяють формуванню пропозиції щодо вирішення проблемних питань.

Основа функціонування будь-якого стартап-проекту – організація надходження в систему інформації від зовнішніх джерел і регулярна актуалізація цієї інформації

Функціональність інформаційної складової декомпозирована на функціональні підсистеми, функції і підфункції. Тому процес реалізації стартап-проекту зі створення платформи розроблення програмного забезпечення

позаштатними працівниками складається з елементів:

– збору, обробки та зберігання інформації;

– аналізу інформації;

– формування інформаційно-аналітичних матеріалів;

– подання інформації;

– захисту інформації.

Процедури та операції (автоматизовані, частково автоматизовані, ручні), пов'язані з предметною областю, слід розділити на п'ять груп:

– операції збору, попередньої обробки (погодження),

– накопичення і зберігання інформаційних ресурсів;

– операції дослідження, аналізу та прогнозування процесів діяльності зі створення стартап-проекту;

– операції розробки та оформлення інформаційно аналітичних матеріалів на основі результатів проведеного аналізу;

– операції надання зацікавленим особам інформації за показниками стартап-проекту.

Результуючі інформаційно-аналітичні матеріали повинні відповідати наступним вимогам:

– актуальності – адекватного відображення стану досліджуваного об'єкта предметної області в кожен момент часу;

– достовірності – відповідності використовуваної при підготовці інформації реальному стану досліджуваних проблем;

– доступності – надання користувачу доступу до інформації за розробленими інформаційно-аналітичними матеріалами;

– лаконічності, ясності і точності викладу;

– наочності – можливості повноцінного візуального сприйняття аналітичного документа як на екрані персонального комп'ютера споживача, так і в паперовому (текстовому) поданні;

– оперативності – можливості підготовки інформаційно-аналітичних матеріалів у встановлені терміни і їх своєчасного доведення до користувача;

– звітності – можливості документування аналітичних даних в установленому вигляді (електронному, письмовому тощо.);

– офіційності – наявності необхідних реквізитів розробника і виклад інформаційно-аналітичних матеріалів в офіційно-діловому стилі.

Таблиця 5.2 – Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технології	Доступність технології
Платформа розроблення програмного забезпечення позаштатними працівниками	Asp.net mvc C# .NET RabbitMQ	Технології наявні, їх потрібно обєднати між собою	Технології доступні
Обрані технології реалізації проекту: Asp.net mvc, C#, .NET, RabbitMQ			

Проект представляє собою клієнт серверне застосування що потребує досить великих витрат ресурсів на створення, оскільки в ньому використовуються нові технології програмування якими володіє невелика частка розробників, але проект цілком реалізуємий.

Аналіз ринкових можливостей запуску стартап-проекту.

Таблиця 5.3 – Попередня характеристика потенційного ринку стартап-проекту.

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, грн/ум.од	10000000000000
3	Динаміка ринку (якісна оцінка)	Зростає

4	Наявність обмежень для входу (вказати характер обмежень)	−
5	Специфічні вимоги до стандартизації та сертифікації	−
6	Середня норма рентабельності в галузі (або по ринку), %	16%

За результатами аналізу проект є досить привабливим для входження оскільки має досить велику рентабельність на ринку розроблення ПЗ і є стартапом в галузі інформаційних технологій що додає йому потенціалу.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту.

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Розроблення програмного забезпечення	Ринки повязаня з обчислювальною технікою	−	Довіра, якість, стабільність роботи

Проект представляє собою платформу яка може бути використаною при розробленні будь-яких програмних продуктів, а на сьогоднішній день без програмного забезпечення не обходиться ні одна промислова галузь, тому потенційний ринок для даного стартапу є величезним, але це лише потенційний ринок, а основний це нові клієнти галузі інформаційних технологій.

Таблиця 5.5 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. монополія	Єдиний проект на ринку	Покращення всіх аспектів які стосуються програмного забезпечення
2. національний	Програмне забезпечення розповсюджується	Покращення всіх аспектів які стосуються програмного

	через інтернет	забезпечення
3. міжгалузева	Унікальний продукт який не конкурує з іншими галузями	Покращення всіх аспектів які стосуються програмного забезпечення

Продовження таблиці 5.5 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
4. Конкуренція за видами товарів:- товарно-родова	Продукт є покращенням вже існуючого виду діяльності	Покращення всіх аспектів які стосуються програмного забезпечення
5. За характером конкурентних переваг нецінова	Перевагами є новий спосіб використання існуючих технологій	Покращення всіх аспектів які стосуються програмного забезпечення
6. За інтенсивністю не марочна	Нова технологія яка не потребує марочної інтенсивності	Покращення всіх аспектів які стосуються програмного забезпечення

На основі проведеного ступеневого аналізу ринку стартап проект має дуже високі шанси зайняти ще не зайняту нішу в галузі розроблення і супроводження ПЗ і стати першим в своєму роді, але конкуренція в нього є у вигляді компаній в яких є уже велика сформована база клієнтів.

Таблиця 5.6 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	–	–	Платформа розроблення програмного забезпечення	Користувачі обчислювальної техніки	Розробники програмного забезпечення
Висновки:	−	Є хороша	Постачальники	Клієнти не	Товари

		можливість виходу на ринок в найблищий час	не диктують умови роботи на ринку	диктують умови роботи на ринку	замінники з часом не витримають конкуренції

На основі проведеного аналізу конкуренції в галузі за М. Портером було виявлено, що на даний момент в розробляємого стартап проекту немає прямих конкурентів, але є потенційні конкуренти у вигляді великих компаній що розробляють ПЗ

Таблиця 5.7 – Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
Монополія	Подібного продукту немає

Найбільшим фактором конкурентоспроможності є той факт що такого продукту немає на ринку при тому що ринок його потребує. І при виході на ринок є всі шанси отримати тимчасову монополію на ринку доки не зявляться значні конкуренти.

Таблиця 5.8 – SWOT- аналіз стартап-проекту

Сильні сторони: монополія	Слабкі сторони:нова технологія
Можливості: революція на ринку розроблення програмного забезпечення	Загрози:відсутність фінансування

Після проведення SWOT аналізу можна виділити той факт що стартап проект має як сильні сторони так і слабкі, проект представляє собою досить новітнє технологічне рішення і його може бути складно просувати на ринок, а за відсутності фінансування хтось інший може заняти нішу на ринку

Таблиця 5.9 – Альтернативи ринкового впровадження стартап-проекту

Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Створення та просування продукту самостійно, без інвесторів	0%	2 роки

На основі вище наданих даних можна зробити висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію, якої майже немає, оскільки продукт зовсім новітній, що надає надає величезну конкурентну перевагу. Найбільш простий метод оцінки стартапу – це знайти оціночну вартість стартапа, порівнюючи його з іншими. Основний принцип цього методу полягає в оцінці стартапу в порівнянні з іншими схожими бізнесами. Наприклад, для технологічного стартапу, який збирається продавати програмне забезпечення для бізнесу, метод порівняння передбачає вивчення стартапів в цьому ж секторі, що пропонують, можливо, схожі продукти, і розгляд того, як їх оцінка співвідноситься з вартістю досліджуваного стартапу. Імплементація системи самому є досить вдалим вибором, якщо знайти фінансування не вдасться.

Таблиця 5.10 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Тестувальники	готові	високий	висока	легко
2	Менеджери	готові	високий	висока	легко
3	Розробники	готові	високий	висока	легко
Які цільові групи обрано: розробники, менеджери, тестувальники					

Оскільки платформа працює з усім ринком розроблення програмного забезпечення, потрібно використовувати масовий маркетинг, що дозволить розповсюдити проект серед компаній що розробляють та супроводжують програмне

забезпечення по всьому світу. Особливо ефективним буде використання контекстної реклами, яка буде показуватися розробникам ПЗ, та іншим працівникам галузі інформаційних технологій, це можливо завдяки збиранню метаданих про користувачів різних інтернет ресурсів пов'язаних з інформаційними технологіями та стартап-проектами.

Таблиця 5.11 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
Самостійна розробка	Масовий маркетинг	Економія коштів	Стратегія спеціалізації

Один з варіантів розвитку даного стартап проекту є самостійна розробка та впровадження платформи на ринок. Це дозволить зекономити значну кількість коштів, але вимагає багато часу, що може привести до появи схожого рішення на ринку. Використання стратегії спеціалізації підходить до даного проекту, оскільки проект займе нішу в галузі інформаційних технологій.

Таблиця 5.12 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Так	Забирати існуючих	Всі	Стратегія лідера

Для того щоб стартап розвивався успішно потрібно використовувати стратегію лідера, впроваджуючи нові технічні рішення раніше за конкурентів, це

можливо завдяки тому що це перший в своєму роді продукт і в нього буде певний час до появи конкурентів щоб зафіксувати своє місце на ринку. Цю стратегію можна відслідкувати якщо поглянути на лідерів ринку галузі інформаційних технологій. Зазвичай той хто перший впровадив технологію маю дуже велику перевагу і конкуренти роками не можуть наздогнати лідера ринку.

Таблиця 5.13 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Легкість користування	Стратегія спеціалізації	Економія коштів	Якісний продукт, зрозумілий у створенні, використанні, супроводженні.

Розроблення маркетингової програми стартап проекту.

Першим кроком є формування маркетингової концепції товару, який отримає споживач представленої в таблиці 5.14

Таблиця 5.14 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Якісне ПЗ	Якісне ПЗ	Менша ціна створення ПЗ
2	розробка макетів ПЗ	розробка макетів ПЗ	Менша ціна створення ПЗ
3	Можливість безетапного створення ПЗ	Можливість безетапного створення ПЗ	Користування платформою не залежно на якому етапі розроблення ПЗ воно знаходиться

Наступним кроком є визначення цінових меж представлених в таблиці 5.15, якими необхідно керуватись при встановленні ціни на товар

Таблиця 15.15 – Визначення меж встановлення ціни

Рівень цін	Рівень цін	Рівень доходів	Верхня та нижня межі

на товари-замінники	на товари-аналоги	цільової групи споживачів	встановлення ціни на товар/послугу
Товари замінники мають високий рівень цін	Товарів аналогів немає	Цільова група споживачів має високий рівень доходів	Нижня межа – це найменша ціна рішення найближчого найдешевшого конкурента, верхня межа, це на 1% більша ціна на найближчого найдорожчого конкурента

За методом порівняння можливо підрахувати вартість клієнтської бази. Можна різні індикатори, а також звичайні показники продажів, валового прибутку і так далі.

Метод порівняння має переваги в тому, що в залежності від показника, його можна застосовувати як для попередньої, так і для після інвестиційної оцінки. В ідеальному випадку при проведенні оцінки необхідно додавати відому інформацію про будь-яку іншу компанію в порівнянні з досліджуваною.

Отже, можливо побачити приклад різних відповідних для порівняння аналогів, придатних для розгляду. Найбільш часто метод порівняння використовує співвідношення EV / Sales для розрахунку вартості – вартість підприємства, поділена на річні продажі.

За допомогою цього методу можна отримати найбільш точний результат, якщо відомий річний дохід, хоча можна використовувати навіть передбачуваний грошовий потік або навіть місячний дохід, якщо такі дані доступні.

Щоб підвищити точність методу порівняння, потрібно буде розглянути використання різних мультиплікаторів. Так, замість використання простого співвідношення EV / Sales, можна вивчити співвідношення User / Sales.

Відповідність – це метод оцінки, який фокусується виключно на перевагах інвестора. У інвестора є певні інвестиційні переваги, і при пошуку він буде вибирати стартапи, «відповідні» цим конкретним критеріям.

Стартап-акселератори та інкубатори часто використовують подібний підхід, хоча для стартапів це не обов'язково.

Даний підхід до оцінки є найменш науковим з усіх методів. Він більше про те, з чим зручніше працювати інвестору, а не про те, скільки насправді коштує стартап.

По суті, цей метод присвячений не стільки докладному розгляду кожного стартапа, скільки знаходженню таких, які задовольнять переваги інвестора.

Ще одним методом оцінки стартапів є дисконтований грошовий потік (DCF-метод). Оцінка стартапа може бути зроблена за допомогою прогнозування конкретного грошового потоку. Ідея цього підходу полягає в погляді на стартап, як на володіння будь-яким іншим інвестиційним активом, наприклад, нерухомістю. Необхідно оцінити і дісконтірувати майбутні грошові потоки стартапу, щоб прийти до сьогоднішньої оцінці.

Цей метод не підходить для стартапів з будь-яким типом фінансової звітності. Однак можна використовувати гіпотетичні грошові потоки і в подальшому дисконтувати їх назад в даний, щоб отримати оцінку. Це може працювати в наступних раундах циклу залучення капіталу, коли у вже будуть в наявності певні надійні фінансові показники і більш висока передбачуваність.

Для оцінки стартапу може застосовуватися метод венчурного капіталу, який передбачає поєднання трьох попередніх методів: порівняння, відповідності та грошового потоку.

Для отримання вартості стартапа використовується метод порівняння. Потім використовується отримана оцінка, щоб рушити назад в інвестиційних раундах і провести оцінку, яка буде відповідати перевагам інвестора.

Існує варіант застосувати прямолінійний підхід і обчислити вартість стартапу, ґрунтуючись на деконструкції або реконструкції проекту.

Деконструкція розглядала б ліквідацію проекту, наприклад, у зв'язку з банкрутством. Коротше кажучи, необхідно розглядати активи стартапу, які можна продати, і підрахувати їх вартість. Це можуть бути офісні приміщення або меблі, технології, якими володіє стартап, і так далі.

Якщо ваш стартап ще молодий і не назбирав ніяких активів, то цей метод застосовувати не слід. Деконструкція не бере до уваги вартість бізнес-ідеї або інших подібних нематеріальних активів.

Реконструкція, з іншого боку, підходить до оцінки з точки зору «побудуй це або купи це». Це означає вивчення основних активів стартапу і базування оцінки на

тому, скільки буде коштувати їх відтворення – тобто реконструкція активів.

Наприклад, основним активом стартапу може бути розроблена технологія. Буде потрібна певна кількість людино-годин, щоб її відтворити. Вартість стартапу буде просто дорівнює вартості цих отриманих людино-годин праці, тому можна використовувати модель реконструкції навіть тоді, коли немає ніяких матеріальних активів.

Оскільки дані, з якими ми працюємо, обмежені, має сенс розглядати різні елементи стартапу разом, саме в цьому і полягає метод комбінування, відомий також під назвою Методу Беркус. Він підсумовує цінність різних елементів стартапу, щоб отримати уявлення щодо його поточної вартості.

Метод досить близький до методу порівняння: необхідно 5 ключових критеріїв для створення гарного стартапа і подальше порівняння досліджуваного стартапу з іншими схожими стартапами.

По суті, ви оцінюється можлива вартість стартапа, розглядаючи, що могли б зробити інші, маючи ці конкретні елементи.

Але існує зворотна сторона комбінованого підходу: конкурентні втрати.

Цей метод оцінки стартапа розглядає, які будуть кінцеві фінансові втрати в разі, якби конкурент отримав перевагу від поглинання досліджуваного проекту.

Для оцінки стартап повинен розглядатися як набір окремих сутностей – він фактично розділяється на шматки для дослідження, як буде впливати втрата однієї з частин на весь бізнес.

Наприклад, готується до випуску якесь програмне забезпечення. Тепер за допомогою методу конкурентних втрат можна оцінити фінансовий збиток, який понесе стартап, якщо інший бізнес отримає це програмне забезпечення. Вартість програмного продукту становитиме, таким чином, фінансові втрати, які будуть отримані від його втрати. Фактично, можливо було б використовувати цей метод навіть стосовно до будь-якої команди: наприклад, яких збитків завдасть стартапу переведення співзасновника до конкурента.

Отже, вибір відповідного методу оцінки стартапу буде залежати від типу бізнесу, точки, в якій перебуваєте проект стосовно залучення інвестицій, і

інвесторів, яких приваблює стартап.

Більшість венчурних інвесторів і бізнес-ангелів використовують порівняльний метод і метод венчурного капіталу, в той час як акселератори користуються методом відповідності.

Поточні високі оцінки можуть не бути ознакою чогось іншого, крім як, переоціненого ринку – стартап може не отримати найвищі оцінки, але це може бути показником розсудливості інвесторів. До того ж висока оцінка може стати проблемою, так як породить занадто високий тиск на стартап.

Таким чином, низька або висока оцінна вартість не означає, що стартап оцінений невірно, це просто показник того, скільки він коштує в даний момент часу.

Економіка оцінки бізнесу сильно відрізняється в залежності від типу компанії. Існує декілька методів, як оцінювати стартапи на ранніх стадіях.

1) Зростання: якщо стартап швидко зростає, це означає, що він вирішує проблему на ринку. Або принаймні те, що компанія заповнює прогалину на ринку, і це в підсумку призводить до залучення нових клієнтів. Крім того, фокус на зростанні також допомагає компанії підлаштовуватися під нові потреби клієнтів.

2) Відтік: якщо стартап швидко росте, але постійно втрачає клієнтів, це означає, що пропозиція відповідала попиту, а сам продукт – ні. Є кілька різних способів виміряти відтік, для цього необхідно звернути увагу на дві головні метрики: відтік користувачів і відтік виручки.

3) Валовий прибуток: компанія швидко зростає і їй вдається утримувати клієнтів. Деякі інвестори використовують «правило сорока» в якості корисної метрики для оцінки здоров'я і надійності стартапів. Це можна розрахувати шляхом додавання темпу зростання компанії до валового прибутку. Ідея в тому, що якщо отримано в результаті більше 40%, значить, варто інвестувати в компанію.

На даний момент варто відзначити, що існує один важливий нюанс між поглядами на ці показники з точки зору інвестицій в акціонерний капітал і позикового фінансування.

Слід зазначити, що існує безліч факторів, які визначають низьку або високу оцінку стартап-проекту. Все це допомагає вибудувати загальну картину. Необхідно

використовувати моделі і стратегії для отримання оцінки та ретельно аналізувати отримані дані.

Отже підсумовуючи вищесказане, можна зробити висновок, що розробляєма платформа досить конкурентоспроможна завдяки своїй іноваційності, але для успішного займання місця на ринку платформа повинна бути завжди попереду своїх конкурентів які можуть з’явитися з часом, тому потрібно розподіляти ресурси не тільки на впровадження платформи а й на її вдосконалення. Платформа для розроблення програмного забезпечення позаштатними працівниками включає в себе багато різних підсистем, в яких є конкуренти поодинці і відповідно кожна з підсистем може приносити дохід по такій же моделі як і в конкурентів, але оскільки розробляєма платформа складається з комплексу підсистем, то для отримання доходу є можливість використання системи в цілому, як приклад можна привести рекламу платних програмних бібліотек в залежності від використовуємих в проекті технологій. Наприклад при розробленні ПЗ з використанням технології для створення ПЗ для настільник компютерів на базі ОС Windows рекомендувати користувачу для покупки програмну бібліотеку з набором контролів для використовуємої ним технології.

ВИСНОВКИ

Процес розроблення програмного забезпечення складається з багатьох етапів та характеризується певними складнощами для замовників, пов’язаними з пошуком виконавців та досягненням всіх вимог до розроблюваного продукту. В роботі вирішена актуальна проблема розроблення програмного забезпечення на різних етапах життєвого циклу. Вирішена об’єднанням їх в один єдиний безетапний процес, призначений для використання позаштатними працівниками, як реалізації «концепції програмного сервісу».

Для досягнення мети дослідження, яка полягає у підвищенні якості та ефективності процесу надання програмних послуг, були вирішені наступні завдання:

– досліджено особливості функціонування платформ для розроблення програмного забезпечення позаштатними працівниками;

– проаналізовано існуючі платформи для розробляємої платформи;

– обґрунтовано вибір підходів і технологій для створення платформи розроблення програмного забезпечення;

– наведено характеристику інформаційної складової діяльності учасників платформи для розроблення програмного забезпечення;

– визначено структуру та характеристику платформи для розроблення програмного забезпечення позаштатними працівниками;

– визначено методи та моделі платформи для розроблення програмного забезпечення позаштатними працівниками;

– спроектовано інформаційну модель та базу даних платформи для розроблення програмного забезпечення позаштатними працівниками;

– реалізовано інтерфейс користувача для розробляємої платформи.

– охарактеризовано програмне, технічне та організаційне забезпечення визначеної платформи.

Область діяльності учасників процесу розроблення на базі платформи для

розроблення програмного забезпечення позаштатними працівниками охоплює стандартний життєвий цикл програмного забезпечення (планування, розробку, тестування, модифікацію, супровід) та певні підтримуючі процеси (оплату, інтеграцію, оренду, взаємодію клієнта та провайдера). В процесі дослідження виявлено переваги та недоліки, у порівнянні із аналогами, обґрунтовано доцільність та ефективність застосування платформи розроблення програмного забезпечення позаштатними працівниками.

Реалізована платформа для розроблення програмного забезпечення позаштатними працівниками є сукупністю певних програмних засобів, використання яких призведе до економії часу на роботу з обробки та аналізу вимог до програмного забезпечення, а також дозволить підвищити якісні та кількісні показники процесів розроблення та тестування.

В ході виконання магістерської дисертації було зроблено такі висновки:

– платформа розроблення програмного забезпечення повинна складатися з двох основних підсистем: бази даних та серверного веб-додатку.

– серверна частина платформи для розроблення програмного забезпечення повинна бути реалізована у якості веб-додатку на базі фреймворку ASP.NET MVC.

– платформа для розроблення програмного забезпечення при необхідності повинна мати інтеграцію та синхронізацію з іншими підсистемами, які здійснюють вплив на процеси життєвого циклу розроблення програмного продукту.

– модуль формування вимог до програмного забезпечення, що розроблюється повинно мати структуровану та гнучку форму з можливістю додавання та змінення вимог на всіх етапах розробки.

– замовник програмного забезпечення повинен мати можливість вибору та підсумкової оцінки найманих працівників.

– розробникам програмного забезпечення має гарантуватися виплата за проектами в повному обсязі згідно поставлених умов.

– розробники програмного забезпечення повинні мати можливість долучати до процесу розробки співавторів за власних бажанням.

– впровадження платформи у практичне використання повинно супроводжуватися та підкріплюватися запровадженням відповідного стартап-процесу.

Загальним результатом розробки платформи розроблення програмного забезпечення позаштатними працівниками має стати реалізація відповідного стартап-проекту на базі отриманих результатів дослідження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jira software. Система відстеження помилок Jira [Електронний ресурс] – Режим доступу до ресурсу: https://ru.atlassian.com/software/jira .

2. Веб сервіс для спільної розробки [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/.

3. Огляд системи Comindware Business Application Platform [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/company/trinion/blog/348618/ .

4. Джексон Дж. Проектирование реляционных баз данных для использование с микро-ЭВМ.-М.: Мир, 1991

5. Харрингтон Д. Проектирование объектно-ориентированных баз данных / Д. Харрингтон – М.: ДМК-Пресс, 2012 – 272 с.

6. Кузин А.В. Разработка баз данных в системе Microsoft Access / А.В. Кузин, В.М. Демин. – М.: Инфра-М, 2014 – 224 с.

7. Дейт К. Введение в системы баз данных. - М. : Издательский дом "Вильямс", 2017. - 1328 с.

8. Илюшечкин В.М. Основы использования и проектирования баз данных / В.М. Илюшечкин. – М.: Юрайт, 2016 – 214 с.

9. Райордан Р. Основы реляционных баз данных/Пер. с англ. – М.: Издательско-торговый дом “Русская редакция”, 2001 – 384 с.

10. Конноли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание/Пер. с англ. – М.: Издательский дом “Вильямс”, 2017 – 1440 с.

11. Грофф Джеймс Р. SQL. Полное руководство / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. – М.: Издательский дом “Вильямс”, 2014 – 960 с.

12. Кузнецов С.Д. Базы данных / С.Д. Кузнецов –СПб.: Academia, 2012. – 496 с.

13. Обломов, С.А. Технологии разработки программного обеспечения / С. А. Обломов. - Санкт-Петербург: Питер, 2002. – 464 с.

14. Ноубл Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с.

15. Коваленко В.В. Проектирование информационных систем / В.В. Коваленко. – М.: Форум, 2014 – 320 с.

16. Экспертные системы в САПР: Лаб. раб. / Сост.: А.А. Кузнецов, О.П. Федосов. Тамбов: Тамб. гос. техн. ун-т. 1995. 33 с.

17. Андерсон Д. Kanban: Successful Evolutionary Change for Your Technology Business / Девід Андерсон., 2010. – 278 с. – (978-0984521401)

18. Кніберг Г. Scrum and XP from the Trenches / Генрік Кніберг., 2007. – 142 с. – (2-ге). –(978-1-4303-2264-1).

19. Концепція програмного сервісу [Электронный ресурс]. Режим доступу https://iconfs.net/infocom2017/kontseptsiya-programnogo-servisu.

20. “Upwork” - глобальний майданчик з пошуку роботи й низку програмних продуктів для роботодавців, які хочуть винаймати й керувати віддаленими спеціалістами [Електронний ресурс] – Режим доступу до ресурсу: https://www.upwork.com/

21. Application programing interface [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/a/application-programming-interface.asp

22. Microsoft Office 365 [Електронний ресурс] – Режим доступу до ресурсу: https://products.office.com/en\-us/office\-365\-personal.

23. Классика баз данных [Електронний ресурс] – Режим доступу до ресурсу: http://citforum.ru/database/classics/

24. Проектирование систем регистрации и анализа данных [Електронний ресурс] – Режим доступу до ресурсу: http://citforum.ru/database/articles/reg_data.shtml

25. Что такое ORM [Електронний ресурс] – Режим доступу до ресурсу: http://www.rte1.ru/sovety/chto/orm/index.html

26. Object-relational mapping [Електронний ресурс] – Режим доступу до ресурсу: http://ru.wikipedia.org/wiki/ORM

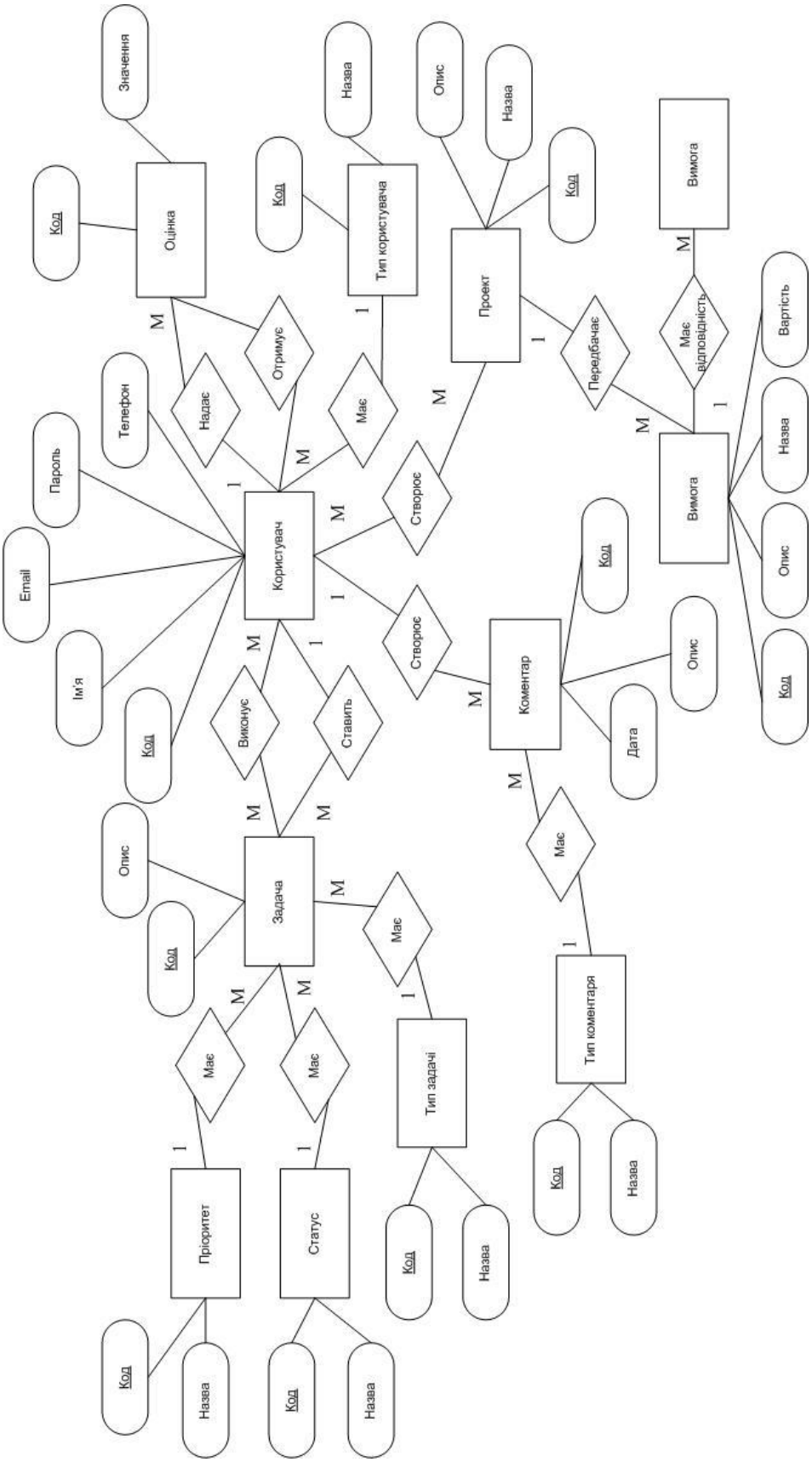
27. Modeling Object/Relational Databases [Електронний ресурс] – Режим доступу до ресурсу: http://ftp.linux.kiev.ua/pub/docs/database/dbmsdigest/dig_1803.html#1

28. ADO.NET Entity Framework [Електронний ресурс] – Режим доступу до ресурсу: http://msdn.microsoft.com/ru-ru/library/bb399572.aspx

29. DataObjects.Net [Електронний ресурс] – Режим доступу до ресурсу: http://mescontrol.ru/features/do.html

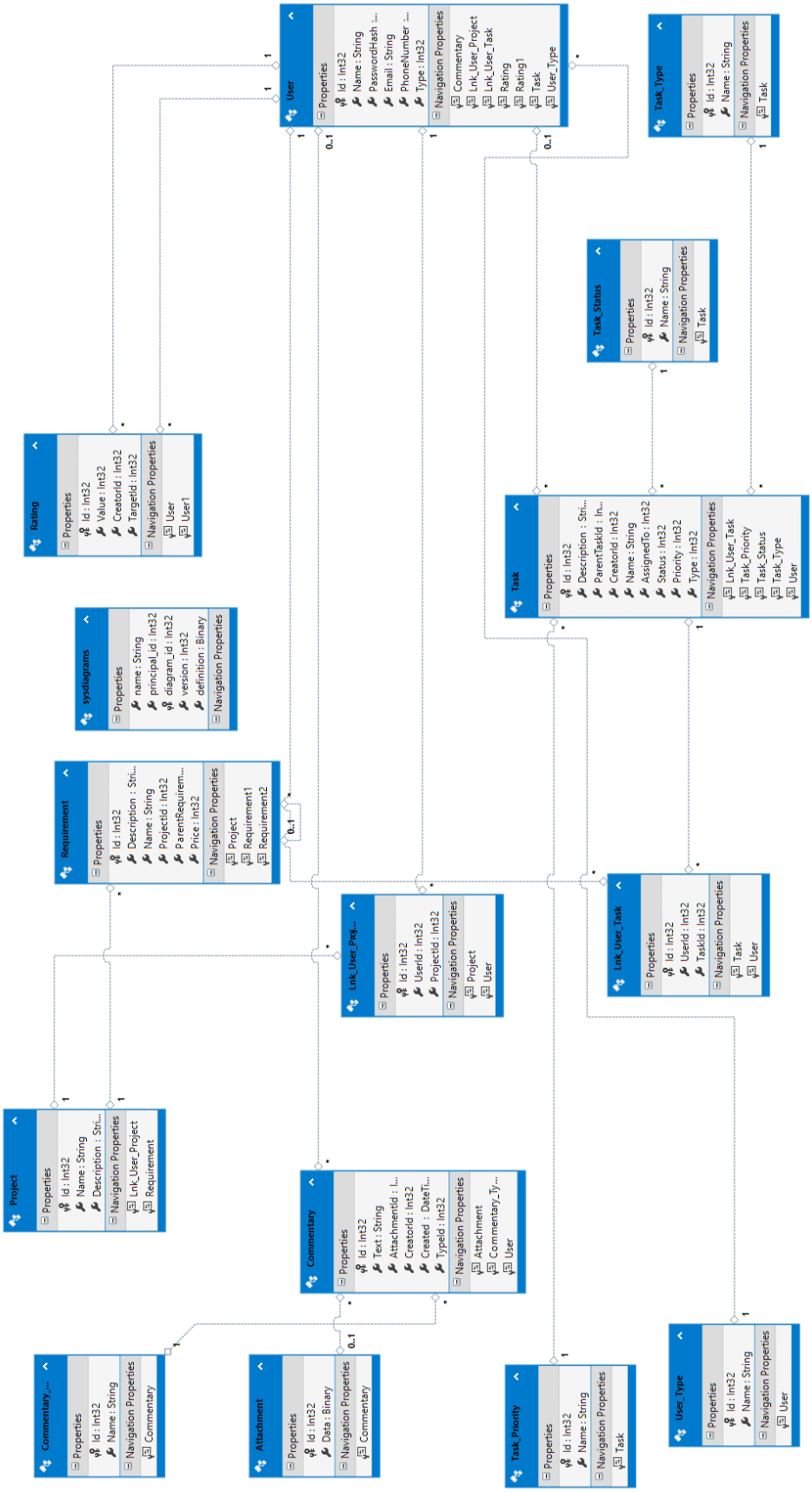
ДОДАТОК А

ER-Діаграма



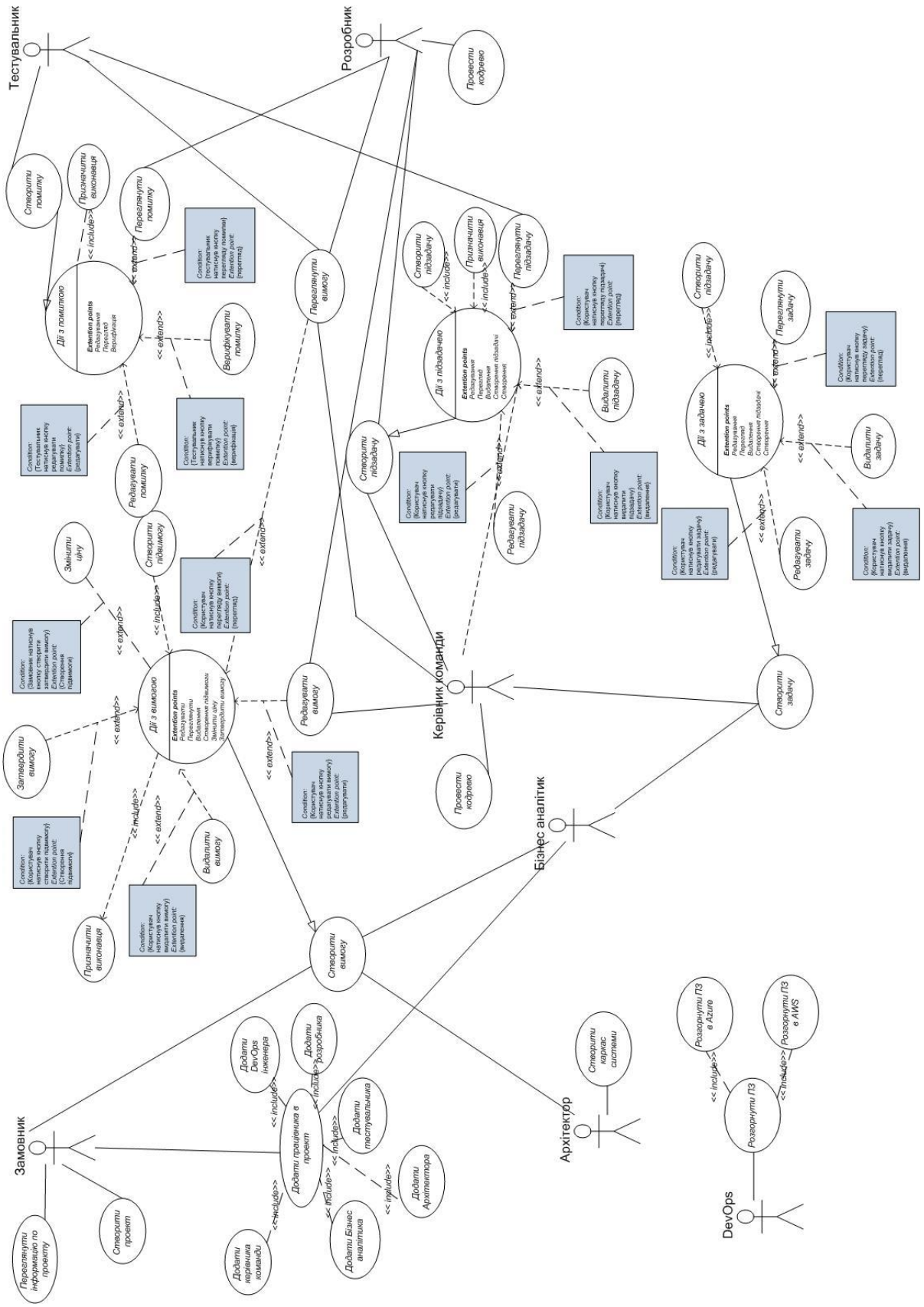
ДОДАТОК Б

Діаграма бази даних



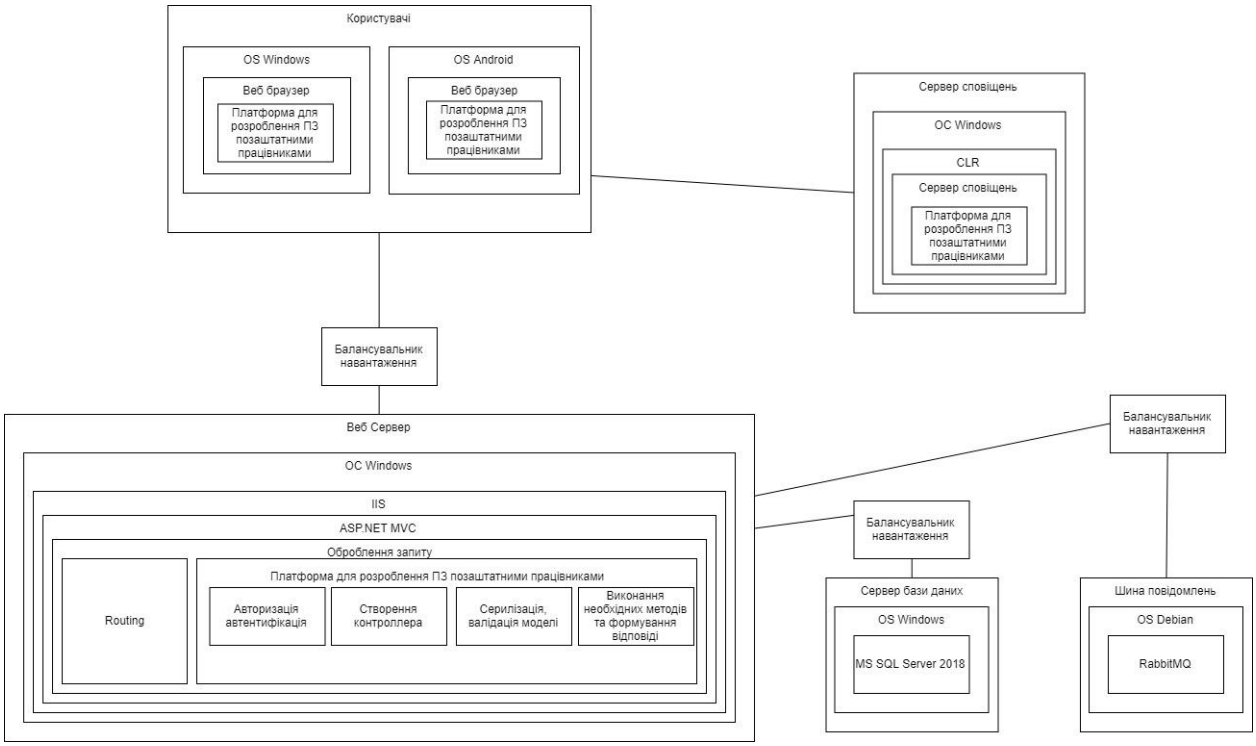
ДОДАТОК В

Діаграма прецедентів



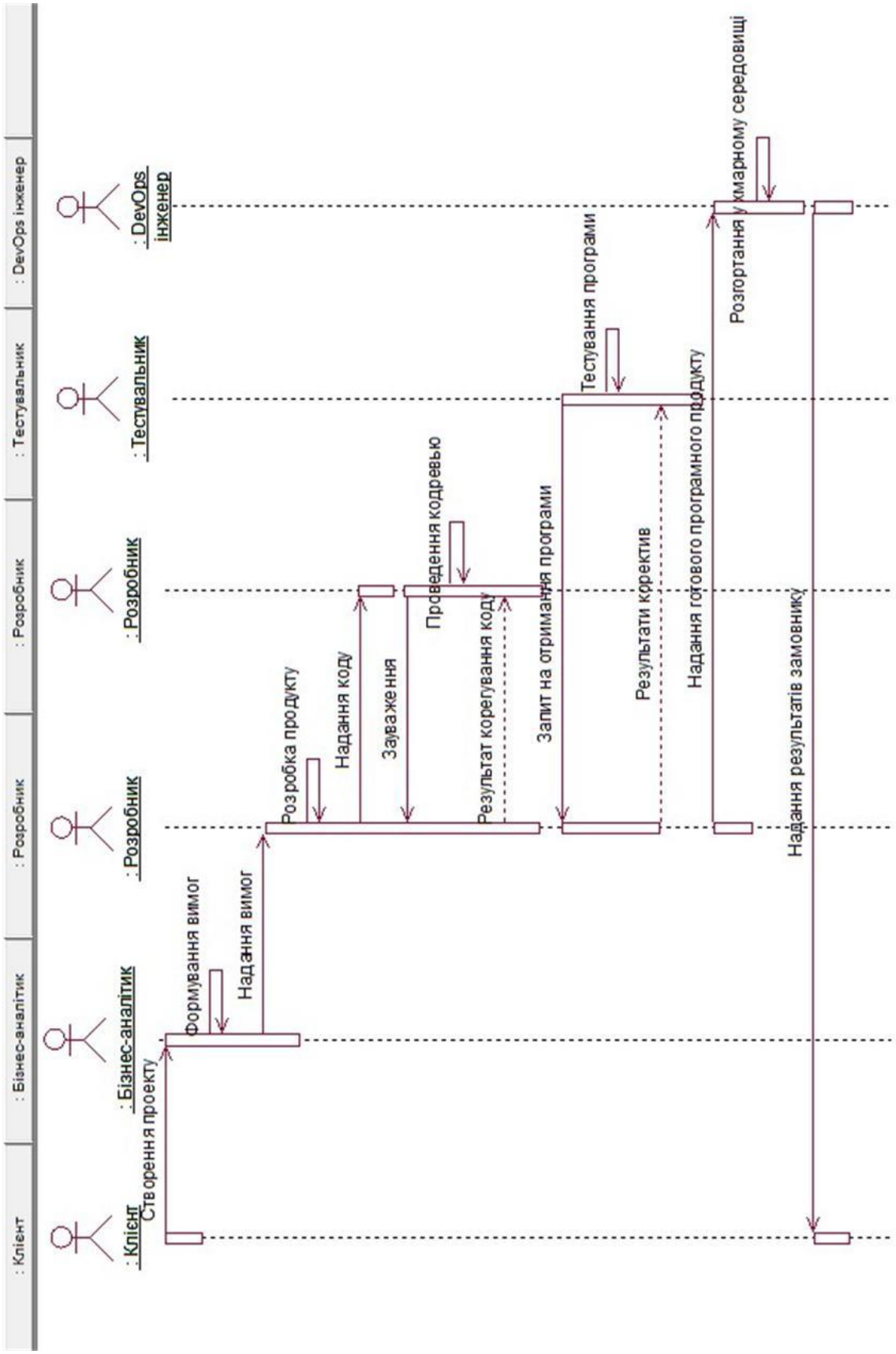
ДОДАТОК Г

Структурна схема


Користувачі
OS Windows
Веб браузер
Платформа для розроблення ПЗ позаштатними працівниками
OS Android
Веб браузер
Платформа для розроблення ПЗ позаштатними працівниками
Сервер сповіщень
ОС Windows
CLR
Сервер сповіщень
Платформа для розроблення ПЗ позаштатними працівниками
Балансувальник навантаження
Веб Сервер
ОС Windows
IIS
ASP.NET MVC
Оброблення запиту
Routing
Платформа для розроблення ПЗ позаштатними працівниками
Авторизація автентифікація
Створення контроллера
Серилізація, валідація моделі
Виконання необхідних методів та формування відповіді
Балансувальник навантаження
Балансувальник навантаження
Сервер бази даних
OS Windows
MS SQL Server 2018
Шина повідомлень
OS Debian
RabbitMQ

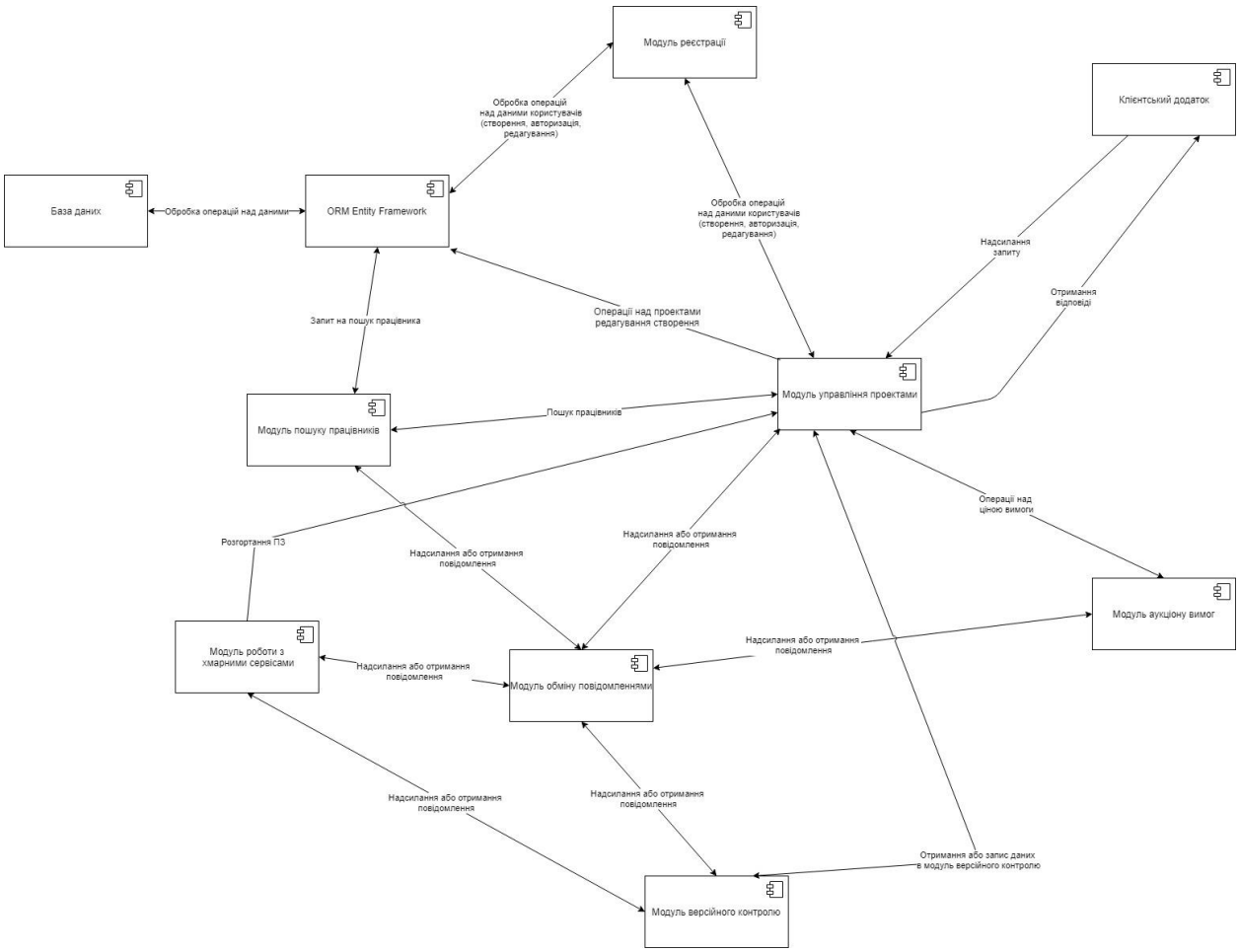
ДОДАТОК Д

Діаграма послідовності



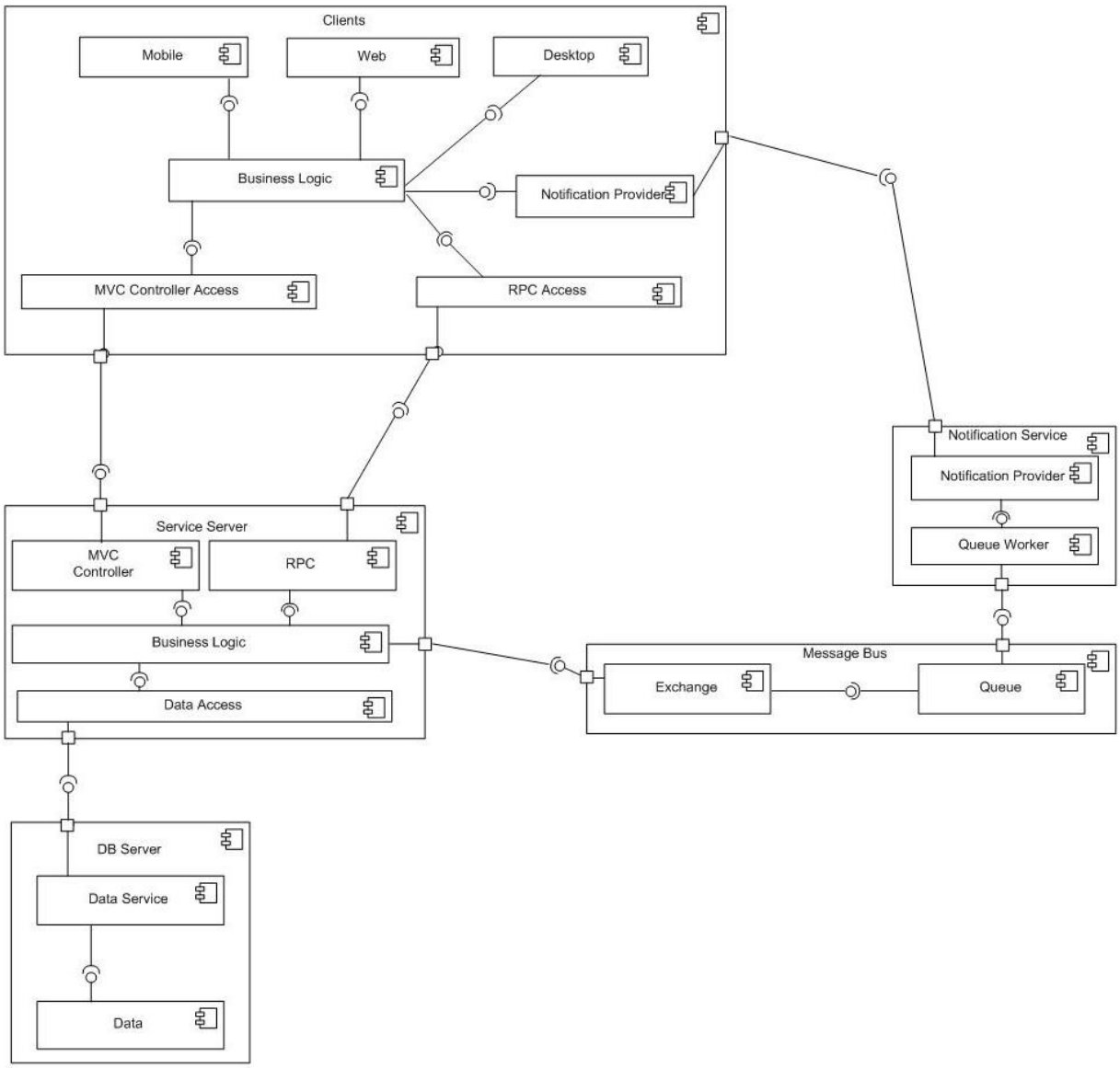
ДОДАТОК Е

Діаграма компонентів


Модуль реєстрації
Клієнтський додаток
Обробка операцій над даними користувачів (створення, авторизація, редагування)
База даних
Обробка операцій над даними
ORM Entity Framework
Обробка операцій над даними користувачів (створення, авторизація, редагування)
Надсилання запиту
Отримання відповіді
Запит на пошук працівника
Операції над проектами редагування створення
Модуль управління проектами
Модуль пошуку працівників
Пошук працівників
Операції над цією вимоги
Розгортання ПЗ
Надсилання або отримання повідомлення
Надсилання або отримання повідомлення
Модуль аукціону вимог
Модуль роботи з хмарними сервісами
Надсилання або отримання повідомлення
Надсилання або отримання повідомлення
Модуль обміну повідомленнями
Надсилання або отримання повідомлення
Надсилання або отримання повідомлення
Отримання або запис даних в модуль версійного контролю
Модуль версійного контролю

ДОДАТОК Ж

Діаграма розгортання



ДОДАТОК И

Діаграма класів

ДОДАТОК К

Публікація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНСТИТУТ МОДЕРНІЗАЦІЇ ЗМІСТУ ОСВІТИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

SUMMER INFOCOM ADVANCED SOLUTIONS 2017

МАТЕРІАЛИ

IV МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

CONFERENCE PROCEEDINGS

IV SCIENTIFIC AND PRACTICAL CONFERENCE

КИЇВ, УКРАЇНА
1-2 ЧЕРВНЯ 2017 РОКУ

УДК 004

Summer InfoCom 2017: Матеріали IV Міжнародної науково-практичної конференції, м. Київ, 1-2 червня 2017 р. – К.: Вид-во ТОВ «Інжиніринг», 2017. – 120 с. – Мови укр., рос., англ.

Конференція входить до Переліку міжнародних та всеукраїнських науково-практичних конференцій здобувачів вищої освіти та молодих учених у 2017 році (додаток до листа Міністерства освіти і науки України № 1/9-24 від 23 січня 2017 року).

Проведення конференції регламентоване наказом ректора КПІ ім. Ігоря Сікорського № 3-183 від 10 травня 2017 р.

Редакційна колегія дотримується прийнятих міжнародною спільнотою принципів публікаційної етики, відображених, зокрема, в рекомендаціях Комітету з етики наукових публікацій (Committee on Publication Ethics, COPE), а також враховує досвід авторитетних міжнародних видавництв. Щоб уникнути недобросовісної практики в публікаційній діяльності (плагіат, виклад недостовірних відомостей та ін.), з метою забезпечення високої якості наукових публікацій, визнання громадськістю отриманих автором наукових результатів, кожен член редакційної колегії, автор, рецензент, видавець, а також установи, які беруть участь в видавничому процесі, зобов'язані дотримуватися етичних стандартів, норм і правил та вживати всіх можливих заходів для запобігання їх порушень. Дотримання правил етики наукових публікацій усіма учасниками цього процесу сприяє забезпеченню прав авторів на інтелектуальну власність, підвищенню якості видання і виключення можливості неправомірного використання авторських матеріалів в інтересах окремих осіб.

ISBN 978-966-2344-54-7

ЗМІСТ

Концепція сервісу програмного забезпечення

Хмелюк Володимир Сергійович
ст. викладач
КПІ ім. Ігоря Сікорського
Україна, Київ

Майер Ілля Сергійович
студент
КПІ ім. Ігоря Сікорського
Україна, Київ

Озеракін Микита Дмитрович
студент
КПІ ім. Ігоря Сікорського
Україна, Київ

Тимошенко Олександр Олександрович
студент
КПІ ім. Ігоря Сікорського
Україна, Київ

Тези доповіді містять опис концепції програмного сервісу, як безперервного та безетапного еволюційного процесу надання програмних послуг, що замінює стандартний життєвий цикл програмного забезпечення (планування, розробку, тестування, модифікацію, супровід) та підтримуючі процеси (оплату, інтеграцію, оренду, взаємодію клієнта та провайдера).

Ключові слова: програмне забезпечення, розробка, сервіс, провайдер, замовник

Завдяки планомірному розвитку та досягненням технічного прогресу багато промислових галузей переходять від продуктових політик до сервісних. Так, наприклад, замість того, щоб купити супутниковий телефон у певної компанії, вкласти з нею договір та оплачувати щомісячну абонплату (продуктовий підхід) – ви маєте змогу купити телефон будь-якого виробника на ваш розсуд, обрати компанію, що буде вам надавати послугу зв'язку (сервіс) і надалі купувати додаткові послуги у цієї компанії за необхідності, орієнтуючись лише на власні потреби та товщину гаманця. Або ж замість купувати собі електрогенератор, ви можете встановити собі лічильник використання електроенергії і платити за сервіс надання доступу до електричних мереж міста згідно обсягів використаної електроенергії та тарифу. Ваш автомобіль теж ймовірно обслуговується в сервісному відділі певного автосалону. Подібних прикладів існує безліч.

І, хоча сфера інформаційних технологій розвивається найбільш динамічно та зазвичай є форпостом запровадження різноманітних нововведень та новаторських рішень, з переходом від продуктового підходу до сервісного не все так гладко. Так в частині «хард» давно вже практикується надання сервісів оренди серверних потужностей, сервісів хмарних містилищ та хмарних обчислень, сервісів послуг інтернет-провайдерів, тощо. А от в частині «софт», тобто програмного забезпечення, – повна тиша. Ви маєте змогу лише купити програмний продукт, або замовити його розробку, що по суті одне й те ж. Правда деякі великі компанії розробники ПЗ, наприклад Microsoft, все ж намагаються запровадити ідею надання послуг використання ПЗ (той же Office 365), але такі виключення поодинокі і лише підтверджують сумне правило.

І, оскільки ще не існує типових сервісів програмного забезпечення[1], спробуємо з'ясувати, якими мають бути такі сервіси і яким вимогам вони мають задовольняти. Отож, основним видом сервісу програмного забезпечення (СПЗ) має стати сервіс безперервної розробки ПЗ (що має замінити собою весь життєвий цикл ПЗ: планування, розробку, тестування, модифікацію, супровід, оплату, тощо). Бо, хоча деякі іменити компанії-розробники наголошують, що вони надають сервіс ПЗ, насправді вони надають лише можливість вибору однієї з продуктових політик використання вже існуючого продукту.

Спробуємо уявити, як би могла виглядати типова взаємодія замовника послуги ПЗ (клієнта сервісу) та розробника ПЗ (в даному контексті провайдера сервісу):

А) Я клієнт:

Я реєструюся на сайті сервісу ПЗ як Замовник. Я описую функціонал, який я хочу отримувати від цільового ПЗ, орієнтовні терміни виходу на робочий режим ПЗ, допустиму вартість умовної години розробки та інші необов'язкові характеристики сервісу. Мою заявку зі статусом «Пошук виконавця» бачать всі зареєстровані розробники ПЗ (провайдери). Зацікавлені провайдери пропонують свої послуги (сервіс ПЗ). Обговорення деталей ведеться тут-же на сайті. В мене є можливість побачити контактні дані провайдерів та зв'язатися з ними безпосередньо для обговорення деталей та нюансів взаємодії. Після вивчення пропозицій я обираю провайдера та замовляю його послугу ПЗ, заключаючи з ним Договір та закриваючи цим свою заявку. З цього моменту мені надається сервіс ПЗ. Я оплачую обумовлену грошову суму на рахунок провайдера, якщо це передбачено (prepaid2). Далі в CS3 я створюю початкові вимоги до ПЗ. Провайдер оцінює мої вимоги в умовних трудовитратах. Я маю змогу деталізувати вимоги, змінити, відмінити, перевпорядкувати, тощо та запропонувати свою вартість реалізації кожної з вимог. Після змін у вимогах і я і провайдер можемо змінити власну запропоновану ціну реалізації вимог. Після певного ітеративного процесу узгодження вимог та їх вартості (так званий аукціон вимог) я підтверджую готовність обраних вимог до реалізації. Провайдер підтверджує факт запуску вимог у розробку.

[1] Під сервісом програмного забезпечення СПЗ надалі будемо розуміти повний комплекс робіт з підтримки життєвого циклу ПЗ в поєднанні з фінансовими аспектами розподілений в часі.

[2] Слово «prepaid» можна перекласти як «передплата». Якщо коротко, prepaid — це спосіб розрахунку з провайдером послуги. Традиційно існують два варіанти оплати надання послуг: кредитний і дебетний. Якщо користуєшся послугою, а потім здійснюєш оплату рахунків що надійшли — це кредитна система. Дебетна система початково передбачає наявність на вашому рахунку певної суми. Наприклад, авансову форму розрахунку можна віднести до дебетових систем: Ви кладете на рахунок певну суму і після цього її витрачаєте

[3] CS – абревіатура референсної програмної системи підтримки СПЗ CleanSlate, що розробляється за участі студентів та викладачів НТУУ «КПІ»

Після реалізації вимоги до ПЗ провайдер повідомляє мене про реалізацію вимоги в новій версії ПЗ, яка автоматично розгортається у мене підсистемою розгортання (згідно налаштувань розгортання та політик підтримки версійності ПЗ). Я підтверджую виконання вимоги в повному обсязі після перевірки функціонування ПЗ. В цей момент з мого рахунку списується оговорена грошова сума а вимога вважається закритою. В процесі моєї взаємодії з провайдером можуть виникати нові вимоги, вимоги можуть ієрархічно розбиватися на дрібніші, об'єднуватися, відмінятися і т.д. згідно алгоритму життєвого циклу вимог. Одними з підвидів вимог будуть вимоги на виправлення помилок, покращення або зміну функціоналу, документування, створення навчальних матеріалів та інші. Надання сервісу провайдером припиняється за двосторонньою згодою, або в випадках передбачених укладеним Договором. Типовим варіантом є нескінченне надання послуги. Переваги, які я отримую від використання концепції програмного сервісу:

- можливість доступу до сервісу ПЗ за допомогою різних робочих обчислювальних машин (ноутбук, смартфон);
- механізм пошуку та конкурсного вибору провайдера сервісу ПЗ;
- середовище для спілкування з провайдером щодо замовленого сервісу ПЗ;
- мінімізацію часу та зусиль на створення технічного завдання на розробку ПЗ, методики перевірок та тестувань та іншої супутньої документації;
- можливість формування поточних версій деяких паперових документів в будь-який час (наприклад ТЗ, звіти, тощо);
- прозорість виконаних робіт провайдером згідно вимогам;
- можливість надгнучкої зміни вимог до ПЗ;
- систему встановлення пріоритетів реалізації вимог узгоджену з провайдером;
- можливість та середовище обліку помилок та відстеження їх виправлення;
- володіння та доступ до містилища поточних напрацювань (вихідні коди, зкомпільовані модулі, документація, тощо);
- можливість використання професійного ревізора оцінок реалізації вимог до ПЗ;
- можливість повного або часткового залучення зацікавлених осіб в процес розробки зі сторони замовника згідно обраної рольової політики в межах надання сервісу ПЗ;
- спрощення розгортання та оновлення ПЗ;
- своєчасність та оперативність оплати робіт виконаних провайдером;
- відстеження та планування бюджету на сервіс ПЗ;
- можливість зміни провайдера в будь-який час без втрати існуючих напрацювань та історії змін вимог до ПЗ;

Б) Я провайдер:

Я реєструюся на сайті сервісу ПЗ як Провайдер. Я вказую профільні предметні області (в яких я маю певні напрацювання та/або конкурентні переваги), посилання на інформацію по реалізованим мною проектам, кваліфікацію та склад команди розробки та супроводження ПЗ, орієнтовну вартість години надання СПЗ, тощо. Мій профіль провайдера разом з рейтингом, відгуками та іншими оцінками бачать як всі зареєстровані клієнти СПЗ так і інші розробники ПЗ (провайдери). Зацікавлені клієнти звертаються до мене з метою уточнення деталей надання СПЗ для можливої подальшої співпраці. Інші провайдери мають змогу звернутися до мене як до співвиконавця з метою доручити мені виконання певних робіт в межах їх власного СПЗ (тобто реалізується платформа пошуку виконавців робіт, але з урахуванням їх планової зайнятості та інших факторів). В мене є можливість побачити контактні дані клієнтів (якщо вони не вказали зворотного) та зв'язатися з ними безпосередньо для обговорення пропозиції по цільовому СПЗ. Після обговорення пропозицій я вкладаю договір на надання СПЗ з клієнтами або іншими провайдерами. Я розпочинаю уточнення початкових вимог замовника до ПЗ за допомогою механізму «аукціон задач» на базовому рівні (тобто до рівня достатнього для планування початкових робіт). Після підтвердження клієнтом починаю виконання робіт по реалізації вказаних вимог. Після виконання вимоги очікую оцінки клієнта на відповідність реалізації. В разі підтвердження клієнтом повного виконання вимоги проводиться фіксація в системі інформації про елемент оплати (фізичне перерахування коштів з таким обліком напряму не зв'язане). Подальші вимоги до ПЗ з'являються в системі в процесі надання СПЗ. Вони можуть генеруватися замовником або розробником з обов'язковим затвердженням зацікавленими сторонами. Переваги, які я отримую від використання концепції програмного сервісу:

- можливість доступу до сервісу ПЗ за допомогою різних робочих обчислювальних машин (ноутбук, смартфон);
- механізм пошуку клієнтів або генеральних підрядників для сервісу ПЗ;
- середовище для спілкування з клієнтами щодо сервісу ПЗ, що надається;
- мінімізацію часу та зусиль на створення технічного завдання на розробку ПЗ, методики перевірок та тестувань та іншої супутньої документації;
- можливість формування поточних версій деяких паперових документів в будь-який час (наприклад звітність по виконанню робіт, часові графіки, іншу інформацію необхідну для подальшого планування та виконання робіт);
- можливість надгнучкої зміни вимог до ПЗ;
- систему встановлення пріоритетів реалізації вимог узгоджену з клієнтом;
- можливість та середовище обліку помилок та відстеження їх виправлення;
- можливість легкого залучення до виконання робіт інших провайдерів СПЗ в якості підрядників;
- спрощення розгортання та оновлення ПЗ;
- можливість передачі клієнта іншому провайдеру (за згоди клієнта) без втрати існуючих напрацювань та історії змін вимог до ПЗ;

- відстеження та часткове прогнозування фінансових надходжень;
- гарантування оплати виконаних робіт клієнтом незалежно від факту повної реалізації вимог у випадку зміни вимог клієнтом, або при припиненні робіт з ініціативи клієнта;
- гнучкі механізми прогнозування та відстеження часових термінів реалізації вимог, задач, робіт, тощо;
- мінімізація бюрократичних витрат;
- значне підвищення взаєморозуміння з клієнтом та максимальне уникнення конфліктних ситуацій;
- реалізація процедур та механізмів можливого вирішення конфліктів за допомогою залучення третьої сторони;
- середовище тісної інтеграції зовнішніх систем розробки та супроводження ПЗ клієнтів в спеціалізовані АРМ провайдера;

Впровадження концепції сервісу програмного забезпечення в життя передбачає проведення низки організаційних заходів, розробки нових та уточнення існуючих методологій розробки та супроводження ПЗ, а також створення програмного забезпечення для підтримки середовища функціонування бізнес-площадок надання СПЗ.

Першою референсною системою підтримки СПЗ є програмна система CleanSlate, що розробляється за участі студентів та викладачів НТУУ «КПІ» та буде детально розглянута в низці наступних тематичних публікацій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Microsoft Office 365 [Електронний ресурс] – Режим доступу до ресурсу: https://products.office.com/en-us/office-365-personal.

2. Лармен К. Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum / Креіг Лармен., 2008. – 368 с. – (1-ше). – (978-0321480965).

3. Андерсон Д. Kanban: Successful Evolutionary Change for Your Technology Business / Девід Андерсон., 2010. – 278 с. – (978-0984521401)

4. Кніберг Г. Scrum and XP from the Trenches / Генрік Кніберг., 2007. – 142 с. – (2-ге). – (978-1-4303-2264-1).

ДОДАТОК Л

Значимий вихідний код

```
USE [master]

GO

/****** Object:  Database [ProjectX_Db]    Script Date: 04.11.2018 23:58:01 ******/

CREATE DATABASE [ProjectX_Db]

 CONTAINMENT = NONE

 ON  PRIMARY

( NAME = N'ProjectX_Db', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\ProjectX_Db.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

 LOG ON

( NAME = N'ProjectX_Db_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\ProjectX_Db_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

ALTER DATABASE [ProjectX_Db] SET COMPATIBILITY_LEVEL = 140

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [ProjectX_Db].[dbo].[sp_fulltext_database] @action = 'enable'

end

GO

ALTER DATABASE [ProjectX_Db] SET ANSI_NULL_DEFAULT OFF

GO

ALTER DATABASE [ProjectX_Db] SET ANSI_NULLS OFF

GO

ALTER DATABASE [ProjectX_Db] SET ANSI_PADDING OFF

GO

ALTER DATABASE [ProjectX_Db] SET ANSI_WARNINGS OFF

GO

ALTER DATABASE [ProjectX_Db] SET ARITHABORT OFF

GO

ALTER DATABASE [ProjectX_Db] SET AUTO_CLOSE OFF

GO

ALTER DATABASE [ProjectX_Db] SET AUTO_SHRINK OFF

GO

ALTER DATABASE [ProjectX_Db] SET AUTO_UPDATE_STATISTICS ON

GO

ALTER DATABASE [ProjectX_Db] SET CURSOR_CLOSE_ON_COMMIT OFF

GO
```

```
ALTER DATABASE [ProjectX_Db] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [ProjectX_Db] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [ProjectX_Db] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [ProjectX_Db] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [ProjectX_Db] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [ProjectX_Db] SET  DISABLE_BROKER
GO
ALTER DATABASE [ProjectX_Db] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [ProjectX_Db] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [ProjectX_Db] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [ProjectX_Db] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [ProjectX_Db] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [ProjectX_Db] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [ProjectX_Db] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [ProjectX_Db] SET RECOVERY FULL
GO
ALTER DATABASE [ProjectX_Db] SET  MULTI_USER
GO
ALTER DATABASE [ProjectX_Db] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [ProjectX_Db] SET DB_CHAINING OFF
GO
ALTER DATABASE [ProjectX_Db] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [ProjectX_Db] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO
ALTER DATABASE [ProjectX_Db] SET DELAYED_DURABILITY = DISABLED
GO
```

```
EXEC sys.sp_db_vardecimal_storage_format N'ProjectX_Db', N'ON'

GO

ALTER DATABASE [ProjectX_Db] SET QUERY_STORE = OFF

GO

USE [ProjectX_Db]

GO

/****** Object:  Table [dbo].[Attachment]    Script Date: 04.11.2018 23:58:01 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Attachment](

	[Id] [int] NOT NULL,

	[Data] [varbinary](50) NULL,

 CONSTRAINT [PK_Attachment] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Commentary]    Script Date: 04.11.2018 23:58:01 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Commentary](

	[Id] [int] NOT NULL,

	[Text] [varchar](50) NOT NULL,

	[AttachmentId] [int] NULL,

	[CreatorId] [int] NULL,

	[Created] [datetime2](7) NOT NULL,

	[TypeId] [int] NOT NULL,

 CONSTRAINT [PK_Commentary] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Commentary_Type]    Script Date: 04.11.2018 23:58:01 ******/
```

```
SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Commentary_Type](

	[Id] [int] NOT NULL,

	[Name] [varchar](50) NOT NULL,

 CONSTRAINT [PK_Commentary_Type] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Lnk_User_Project]    Script Date: 04.11.2018 23:58:01 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Lnk_User_Project](

	[Id] [int] NOT NULL,

	[UserId] [int] NOT NULL,

	[ProjectId] [int] NOT NULL,

 CONSTRAINT [PK_Lnk_User_Project] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Lnk_User_Task]    Script Date: 04.11.2018 23:58:01 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Lnk_User_Task](

	[Id] [int] NOT NULL,

	[UserId] [int] NOT NULL,

	[TaskId] [int] NOT NULL,

 CONSTRAINT [PK_Lnk_User_Task] PRIMARY KEY CLUSTERED

(
```

```
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Project]    Script Date: 04.11.2018 23:58:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Project](
	[Id] [int] NOT NULL,
	[Name] [varchar](50) NOT NULL,
	[Description] [varchar](50) NULL,
 CONSTRAINT [PK_Project] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Rating]    Script Date: 04.11.2018 23:58:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Rating](
	[Id] [int] NOT NULL,
	[Value] [int] NOT NULL,
	[CreatorId] [int] NOT NULL,
	[TargetId] [int] NOT NULL,
 CONSTRAINT [PK_Rating] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Requirement]    Script Date: 04.11.2018 23:58:02 ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Requirement](

	[Id] [int] NOT NULL,

	[Description] [varchar](50) NOT NULL,

	[Name] [varchar](50) NULL,

	[ProjectId] [int] NOT NULL,

	[ParentRequirement] [int] NULL,

	[Price] [int] NULL,

 CONSTRAINT [PK_Requirement] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Task]    Script Date: 04.11.2018 23:58:02 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Task](

	[Id] [int] NOT NULL,

	[Description] [varchar](50) NULL,

	[ParentTaskId] [int] NULL,

	[CreatorId] [int] NOT NULL,

	[Name] [varchar](50) NOT NULL,

	[AssignedTo] [int] NULL,

	[Status] [int] NOT NULL,

	[Priority] [int] NOT NULL,

	[Type] [int] NOT NULL,

 CONSTRAINT [PK_Task] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Task_Priority]    Script Date: 04.11.2018 23:58:02 ******/

SET ANSI_NULLS ON

GO
```

```
SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Task_Priority](

	[Id] [int] NOT NULL,

	[Name] [varchar](50) NOT NULL,

 CONSTRAINT [PK_Task_Priority] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Task_Status]    Script Date: 04.11.2018 23:58:02 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Task_Status](

	[Id] [int] NOT NULL,

	[Name] [varchar](50) NOT NULL,

 CONSTRAINT [PK_Task_Status] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/****** Object:  Table [dbo].[Task_Type]    Script Date: 04.11.2018 23:58:02 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Task_Type](

	[Id] [int] NOT NULL,

	[Name] [varchar](50) NOT NULL,

 CONSTRAINT [PK_Task_Type] PRIMARY KEY CLUSTERED

(

	[Id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]
```

```
GO
/****** Object:  Table [dbo].[User]    Script Date: 04.11.2018 23:58:02 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[User](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[Name] [varchar](50) NOT NULL,
	[PasswordHash] [varchar](50) NOT NULL,
	[Email] [varchar](50) NULL,
	[PhoneNumber] [varchar](50) NULL,
	[Type] [int] NOT NULL,
 CONSTRAINT [PK_User] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[User_Type]    Script Date: 04.11.2018 23:58:02 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[User_Type](
	[Id] [int] NOT NULL,
	[Name] [varchar](50) NOT NULL,
 CONSTRAINT [PK_User_Type] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
INSERT [dbo].[Commentary_Type] ([Id], [Name]) VALUES (1, N'UserMessage')
INSERT [dbo].[Commentary_Type] ([Id], [Name]) VALUES (2, N'TaskCommentary')
INSERT [dbo].[Task_Priority] ([Id], [Name]) VALUES (1, N'Low')
INSERT [dbo].[Task_Priority] ([Id], [Name]) VALUES (2, N'Medium')
INSERT [dbo].[Task_Priority] ([Id], [Name]) VALUES (3, N'High')
INSERT [dbo].[Task_Status] ([Id], [Name]) VALUES (1, N'New')
```

```
INSERT [dbo].[Task_Status] ([Id], [Name]) VALUES (2, N'InProgress')
INSERT [dbo].[Task_Status] ([Id], [Name]) VALUES (3, N'Done')
INSERT [dbo].[Task_Type] ([Id], [Name]) VALUES (1, N'Dev')
INSERT [dbo].[Task_Type] ([Id], [Name]) VALUES (2, N'QA')
INSERT [dbo].[Task_Type] ([Id], [Name]) VALUES (3, N'CodeReview')
SET IDENTITY_INSERT [dbo].[User] ON

INSERT [dbo].[User] ([Id], [Name], [PasswordHash], [Email], [PhoneNumber], [Type]) VALUES (0, N'1', N'1', N'1', N'1', 2)
INSERT [dbo].[User] ([Id], [Name], [PasswordHash], [Email], [PhoneNumber], [Type]) VALUES (1, N'3', N'3', N'3', N'3', 2)
INSERT [dbo].[User] ([Id], [Name], [PasswordHash], [Email], [PhoneNumber], [Type]) VALUES (2, N'4', N'4', N'4', N'4', 2)
INSERT [dbo].[User] ([Id], [Name], [PasswordHash], [Email], [PhoneNumber], [Type]) VALUES (3, N'5', N'5', N'5', N'5', 2)
INSERT [dbo].[User] ([Id], [Name], [PasswordHash], [Email], [PhoneNumber], [Type]) VALUES (4, N'7', N'7', N'7', N'7', 2)
SET IDENTITY_INSERT [dbo].[User] OFF
INSERT [dbo].[User_Type] ([Id], [Name]) VALUES (1, N'Manager')
INSERT [dbo].[User_Type] ([Id], [Name]) VALUES (2, N'Dev')
INSERT [dbo].[User_Type] ([Id], [Name]) VALUES (3, N'QA')
ALTER TABLE [dbo].[Commentary]  WITH CHECK ADD  CONSTRAINT [FK_Commentary_Attachment] FOREIGN KEY([AttachmentId])
REFERENCES [dbo].[Attachment] ([Id])
GO
ALTER TABLE [dbo].[Commentary] CHECK CONSTRAINT [FK_Commentary_Attachment]
GO
ALTER TABLE [dbo].[Commentary]  WITH CHECK ADD  CONSTRAINT [FK_Commentary_Commentary_Type] FOREIGN KEY([TypeId])
REFERENCES [dbo].[Commentary_Type] ([Id])
GO
ALTER TABLE [dbo].[Commentary] CHECK CONSTRAINT [FK_Commentary_Commentary_Type]
GO
ALTER TABLE [dbo].[Commentary]  WITH CHECK ADD  CONSTRAINT [FK_Commentary_User] FOREIGN KEY([CreatorId])
REFERENCES [dbo].[User] ([Id])
GO
ALTER TABLE [dbo].[Commentary] CHECK CONSTRAINT [FK_Commentary_User]
GO
ALTER TABLE [dbo].[Lnk_User_Project]  WITH CHECK ADD  CONSTRAINT [FK_Lnk_User_Project_Project] FOREIGN KEY([ProjectId])
REFERENCES [dbo].[Project] ([Id])
GO
ALTER TABLE [dbo].[Lnk_User_Project] CHECK CONSTRAINT [FK_Lnk_User_Project_Project]
GO
ALTER TABLE [dbo].[Lnk_User_Project]  WITH CHECK ADD  CONSTRAINT [FK_Lnk_User_Project_User] FOREIGN KEY([UserId])
REFERENCES [dbo].[User] ([Id])
GO
ALTER TABLE [dbo].[Lnk_User_Project] CHECK CONSTRAINT [FK_Lnk_User_Project_User]
```

```
GO
ALTER TABLE [dbo].[Lnk_User_Task]  WITH CHECK ADD  CONSTRAINT [FK_Lnk_User_Task_Task] FOREIGN KEY([TaskId])
REFERENCES [dbo].[Task] ([Id])
GO
ALTER TABLE [dbo].[Lnk_User_Task] CHECK CONSTRAINT [FK_Lnk_User_Task_Task]
GO
ALTER TABLE [dbo].[Lnk_User_Task]  WITH CHECK ADD  CONSTRAINT [FK_Lnk_User_Task_User] FOREIGN KEY([UserId])
REFERENCES [dbo].[User] ([Id])
GO
ALTER TABLE [dbo].[Lnk_User_Task] CHECK CONSTRAINT [FK_Lnk_User_Task_User]
GO
ALTER TABLE [dbo].[Rating]  WITH CHECK ADD  CONSTRAINT [FK_Rating_User] FOREIGN KEY([CreatorId])
REFERENCES [dbo].[User] ([Id])
GO
ALTER TABLE [dbo].[Rating] CHECK CONSTRAINT [FK_Rating_User]
GO
ALTER TABLE [dbo].[Rating]  WITH CHECK ADD  CONSTRAINT [FK_Rating_User1] FOREIGN KEY([TargetId])
REFERENCES [dbo].[User] ([Id])
GO
ALTER TABLE [dbo].[Rating] CHECK CONSTRAINT [FK_Rating_User1]
GO
ALTER TABLE [dbo].[Requirement]  WITH CHECK ADD  CONSTRAINT [FK_Requirement_Project] FOREIGN KEY([ProjectId])
REFERENCES [dbo].[Project] ([Id])
GO
ALTER TABLE [dbo].[Requirement] CHECK CONSTRAINT [FK_Requirement_Project]
GO
ALTER TABLE [dbo].[Requirement]  WITH CHECK ADD  CONSTRAINT [FK_Requirement_Requirement] FOREIGN KEY([ParentRequirement])
REFERENCES [dbo].[Requirement] ([Id])
GO
ALTER TABLE [dbo].[Requirement] CHECK CONSTRAINT [FK_Requirement_Requirement]
GO
ALTER TABLE [dbo].[Task]  WITH CHECK ADD  CONSTRAINT [FK_Task_Task_Priority] FOREIGN KEY([Priority])
REFERENCES [dbo].[Task_Priority] ([Id])
GO
ALTER TABLE [dbo].[Task] CHECK CONSTRAINT [FK_Task_Task_Priority]
GO
ALTER TABLE [dbo].[Task]  WITH CHECK ADD  CONSTRAINT [FK_Task_Task_Status] FOREIGN KEY([Status])
REFERENCES [dbo].[Task_Status] ([Id])
GO
ALTER TABLE [dbo].[Task] CHECK CONSTRAINT [FK_Task_Task_Status]
```

```
GO
ALTER TABLE [dbo].[Task]  WITH CHECK ADD  CONSTRAINT [FK_Task_Task_Type] FOREIGN KEY([Type])
REFERENCES [dbo].[Task_Type] ([Id])
GO
ALTER TABLE [dbo].[Task] CHECK CONSTRAINT [FK_Task_Task_Type]
GO
ALTER TABLE [dbo].[Task]  WITH CHECK ADD  CONSTRAINT [FK_Task_User] FOREIGN KEY([AssignedTo])
REFERENCES [dbo].[User] ([Id])
GO
ALTER TABLE [dbo].[Task] CHECK CONSTRAINT [FK_Task_User]
GO
ALTER TABLE [dbo].[User]  WITH CHECK ADD  CONSTRAINT [FK_User_User_Type] FOREIGN KEY([Type])
REFERENCES [dbo].[User_Type] ([Id])
GO
ALTER TABLE [dbo].[User] CHECK CONSTRAINT [FK_User_User_Type]
GO
USE [master]
GO
ALTER DATABASE [ProjectX_Db] SET  READ_WRITE
GO
```